Hindawi
Security and Communication Networks
Volume 2019, Article ID 9684239, 11 pages
https://doi.org/10.1155/2019/9684239

Research Article

A Lightweight BCH Code Corrector of TRNG with Measurable Dependence

Hojoong Park, Yongjin Yeom, and Ju-Sung Kang

Department of Information Security, Cryptology and Mathematics, Graduate School of Financial Information Security, Kookmin University, 77 Jeongneung-ro, Seongbuk-Gu, Seoul 02707, Republic of Korea

Correspondence should be addressed to Ju-Sung Kang; jskang@kookmin.ac.kr

Received 21 January 2019; Accepted 16 April 2019; Published 13 May 2019

Academic Editor: Prosanta Gope

We propose a new lightweight BCH code corrector of the random number generator such that the bitwise dependence of the output value is controllable. The proposed corrector is applicable to a lightweight environment and the degree of dependence among the output bits of the corrector is adjustable depending on the bias of the input bits. Hitherto, most correctors using a linear code are studied on the direction of reducing the bias among the output bits, where the biased input bits are independent. On the other hand, the output bits of a linear code corrector are inherently not independent even though the input bits are independent. However, there are no results dealing with the independence of the output bits. The well-known von Neumann corrector has an inefficient compression rate and the length of output bits is nondeterministic. Since the heavy cryptographic algorithms are used in the NIST's conditioning component to reduce the bias of input bits, it is not appropriate in a lightweight environment. Thus we have concentrated on the linear code corrector and obtained the lightweight BCH code corrector with measurable dependence among the output bits as well as the bias. Moreover, we provide some simulations to examine our results.

1. Introduction

Random number generator (RNG) is essential in the modern cryptography system and used to generate the security parameters such as secret key, initialization vector, nonce, salt, and so on. The random numbers used for cryptographic purposes should be generated by the cryptographically secure random number generator [1]. The cryptographically secure random number generator is composed of the true random number generator (TRNG) and the pseudorandom number generator (PRNG) [1–3] as shown in Figure 1. The nondeterministic outputs are generated by TRNG and are the root of security, also known as the entropy source, for the cryptographically secure random number generator. In PRNG process, the cryptographically secure random numbers are finally generated by a deterministic algorithm using the seed, the output of TRNG, as input value. Hence the security of the cryptographically secure random number generator depends on the nondeterministic entropy source and all its constituent parts [3]. In order to generate the cryptographically secure random number, we first collect the entropy from the noise source such as thermal noise, ring oscillator, noise from quantum effect, outputs of crypto API in operation system, and CPU jitter noise source [4, 5] in TRNG process.

It is difficult to directly utilize the noise source as a seed, the input value of PRNG, since the bias of noise source is able to be exploited to attack the cryptosystem by using the statistical estimation of the next TRNG output bits [6]. To solve this problem, various post-processing components in TRNG are used as shown in Figure 1. The principal role of the post-processing component is reducing the bias of the noise source in TRNG. There are well-known post-processing components such as von Neumann corrector [7], XOR corrector [8], NIST's conditioning component [4], and correctors using linear code and nonlinear code (code corrector). Until now most researches related to the post-processing components have been conducted on the subject of reducing the bias of output bits [6, 9–12]. The outputs of von Neumann corrector [7] are perfectly unbiased if the input bits are independent. The bias of the output bits from XOR corrector [8] is proven to be $2\epsilon^2$, where the bias of input bits is ϵ, for any $0 < \epsilon < 1$, and the input bits are independent. In the NIST's

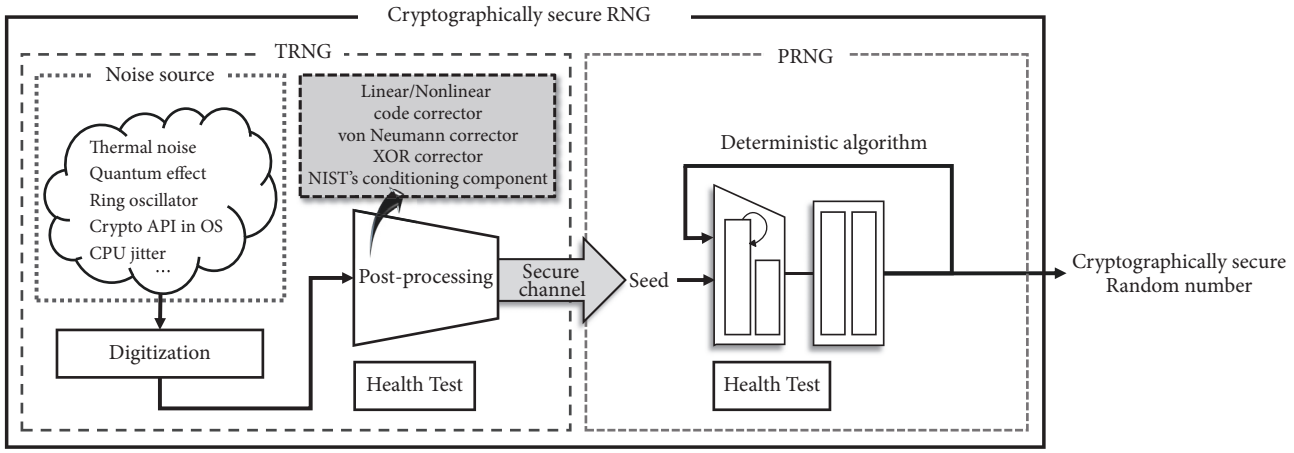

Figure 1: Cryptographically secure random number generator and post-processing components in TRNG.

conditioning component [4], the FIPS-approved or NIST-recommended cryptographic algorithms are used to reduce the bias of input values. The quality of the output bits from the NIST's conditioning component is only empirically ensured by the pseudorandomness and the statistical randomness of the used cryptographic algorithms which are heavy to a lightweight environment. The linear code and nonlinear code are employed as the code corrector for reducing the bias of input bits. The advantage of the code corrector is that the bias of output value is theoretically determined by the bias of input bits and the used code.

Meanwhile, the results of research on the independence of the output bits of post-processing component are known as follows: The output bits of the von Neumann corrector and XOR corrector are proven to be bitwise independent when the input bits are independent and stationary [9]. We concede that the output bits of NIST's conditioning component are heuristically independent since the outputs are generated by heavy cryptographic algorithms. On the other hand, the output bits of the code corrector are intrinsically not independent even though the input bits are independent; however, there are no results dealing with the independence of the output bits. Therefore we inspect the code corrector from the view point of independence in this paper.

Related Works. There are a number of works to design post-processing component using linear code and to study on the direction of reducing the bias of the output bits where the biased input bits are independent and stationary. As a noticeable result, Markovski et al. [10] have proposed a new linear corrector using quasigroup and have analyzed the output of linear corrector using KStest. Then, they have provided some simulation results and mentioned that the linear corrector is efficiently implemented in both hardware and software. Dichtl [11] has refuted the post-processing component of Markovski and proposed a new linear corrector using XOR compression function. He has analyzed the compression function for input bias in order to reduce the output bias. Lacharme [12] has proposed post-processing components using a resilient function and a cyclic code. He has calculated the upper bound of output bias using relation between resilient function and cyclic code and analyzed a bias and minimal entropy of output. Kim et al. [6] have designed a linear corrector overcoming the minimum distance limitation using quadratic residue code. They have analyzed outputs of the corrector using entropy test in AIS.31 standard and mutual information and have provided a hardware architecture. Kwok et al. [9] have compared code corrector and von Neumann corrector using compression rate, output's bias, and adversary bias. They have also provided good ways to implement linear code corrector in hardware based on their analysis of linear code corrector.

Our Contribution. We propose a new lightweight BCH code corrector that bitwise dependence of the output value is adjustable. Since most code correctors have been studied on the subject of reducing the bias of the output bits where the biased input bits are independent, we concentrate on the dependence of the output bits and define a new measure, *degree of dependence*, in order to make the lightweight BCH code corrector with controllable dependence as shown in Figure 2. We obtain the fact that *degree of dependence* is related to the bias of input bits through theoretical and simulation results. Moreover, the [**7**, **4**, **3**] BCH code corrector is applicable to the lightweight environment such as IoT, embedded, and mobile devices because of its small code size. In other words, the proposed BCH code corrector is applicable to the lightweight environment as post-processing component of TRNG, and the dependence among the output bits of this corrector is controllable depending on the bias of the input bits.

Organization. The rest of this paper is organized as follows. In Section 2, we introduce the post-processing components and their properties. In Section 3, we examine the output bits of the BCH code corrector from the view point of mutual independence. In Section 4, we formally define a new measure, *degree of dependence*, and compare another method measuring dependence. In Section 5, we exploit *degree of dependence* in order to measure the dependence of the output

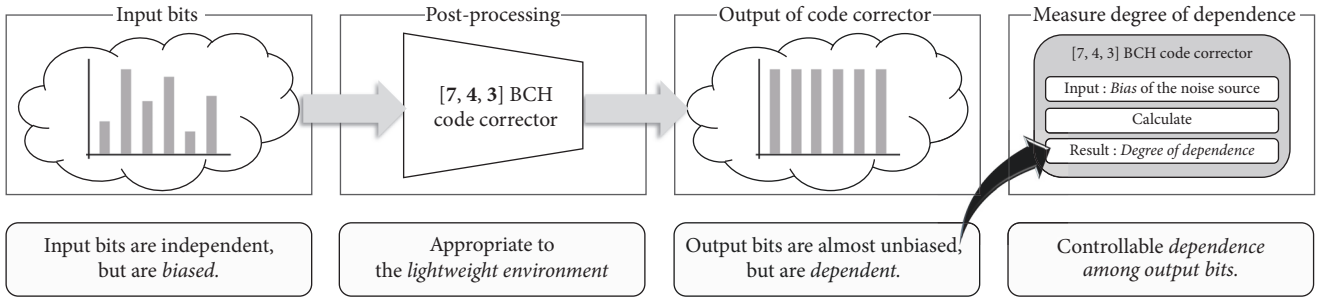

Figure 2: A new lightweight BCH code corrector with controllable dependence.

Table 1: Comparison of post-processing components of TRNG.

	Properties	Limitations
von Neumann corrector	The output is unbiased and independent.	The output length is nondeterministic since compression rate is 0.25 on average.
XOR corrector	The post-processed data is independent and implementation is easy.	Compression rate is fixed as 0.5. The bias of output is fixed as $2\epsilon^2$ if the bias of input is ϵ.
NIST's conditioning component	The output bits of conditioning component are regarded as heuristically unbiased and independent.	The output bits have not been theoretically proven from the viewpoint of unbias and independence.
Code corrector	It is able to adjust the bias of output bits depending on used code and input bias.	The output bits of the code corrector are not independent even though the input sequence is independent.

bits of $[\mathbf{7}, \mathbf{4}, \mathbf{3}]$ BCH code corrector. In Section 6, we provide some experimental results for inspecting our results, and we propose the application of the BCH code corrector of TRNG in a lightweight environment. Finally, Section 7 is the conclusion of this paper.

2. Post-Processing Component

The post-processing component has been utilized in TRNG mainly for decreasing the bias of noise source. The basic role of post-processing component in TRNG is to make seed having high entropy rate, since the security of the cryptographically secure random number generator essentially depends on the entropy of the seed [4]. There are the representative post-processing components applied in TRNG such as von Neumann corrector [7], XOR corrector [8], NIST's conditioning component [4], and code corrector. Some properties and limitations of the post-processing components are summarized in Table 1.

On the other hand, most post-processing components have been studied on the direction of reducing the bias of output bits. The von Neumann corrector generates perfectly unbiased and independent outputs when the input bits are independent. Let x and y be an input bit and output bit, respectively, and let ϵ_{in} and ϵ_{out} be a bias of input bit and a bias of output bit, respectively; then the bias of input bit is defined as $\epsilon_{in} = (1/2)(P(x = 0) - P(x = 1))$. When two input bits are '01', the von Neumann corrector outputs '0' and when two input bits are '10', it outputs '1' in the von Neumann corrector. On the other hand, the other cases such as two bits of noise source are '00' or '11', and there is no output from the von Neumann corrector. Then, the probabilities of the output bit are calculated as $P(y = 0) = P(y = 1) = 1/4 - \epsilon_{in}^2$ due to independence of input bits. Since the bias of output bit is $\epsilon_{out} = (1/2)(P(y = 0) - P(y = 1)) = 0$, the output bit of the von Neumann corrector is perfectly unbiased. However, the von Neumann corrector has some problems that not only the output length is nondeterministic but also the expected output length is one-fourth of the input length since the compression rate is 0.25 on average.

The XOR corrector has properties such that the output bits of XOR corrector are independent when the input bit is independent and stationary, and the XOR corrector is also easy to implement [9]. However, it is not flexible since the compression rate is fixed to one-twice and the fixed output's bias is fixed to $2\epsilon^2$ when input bias is ϵ [8]. In NIST's conditioning component, FIPS-approved or NIST-recommended cryptographic algorithms such as Hash function in FIPS 180-4 [13] or in FIPS 202 [14], HMAC in FIPS 198-1 [15], CBC-MAC in SP 800-90B [4] and so on are used to reduce the bias of input bits. Hence the unbiased outputs are heuristically guaranteed by the pseudorandomness of outputs of the cryptographic algorithms. In addition, because it is difficult to theoretically analyze the conditioning component from the viewpoint of the output's bias and independence, there is no theoretical result as other post-processing functions.

The linear codes such as dual code [6], BCH code [9], and cyclic code [12] are useful to the code corrector. Until now, the code corrector has been actively studied on the direction of reducing the bias of output bits [9–12], since the compression rate and bias of the output bits could be adjusted by the bias of input value and the characteristics of used code. On

Table 2: Distribution of output bit of [7, 4, 3] BCH code corrector.

z	000	001	010	011	100	101	110	111	Sum
$y = 0$	$(1/2+\epsilon)^3$	0	0	$(1/2+\epsilon)\cdot(1/2-\epsilon)^2$	0	$(1/2+\epsilon)\cdot(1/2-\epsilon)^2$	$(1/2+\epsilon)\cdot(1/2-\epsilon)^2$	0	$1/2+4\epsilon^3$
$y = 1$	0	$(1/2-\epsilon)\cdot(1/2+\epsilon)^2$	$(1/2-\epsilon)\cdot(1/2+\epsilon)^2$	0	$(1/2-\epsilon)\cdot(1/2+\epsilon)^2$	0	0	$(1/2-\epsilon)^3$	$1/2-4\epsilon^3$

the other hand, the output bits of the code corrector are intrinsically not independent even though the input bits are bitwise independent; however, there are no results pertaining to the independence of the output bits. Hence we examine the code corrector from the viewpoint of independence in this paper.

3. Mutual Independence of BCH Code Corrector

In this section, we examine the BCH code corrector from the viewpoint of the mutual independence. In order to inspect the properties of BCH code corrector, we choose the $[\mathbf{7}, \mathbf{4}, \mathbf{3}]$ BCH code, the smallest BCH code, and also take the same assumption as in [9–12]: the input bits are bitwise independent, biased, and stationary. We verify the fact that the output bits of BCH code corrector are not independent even though the input bits are bitwise independent.

Let the input bits of noise source be $\mathbf{x} = (x_0, x_1, \ldots, x_{n-1})$ and let the post-processed bits be $\mathbf{y} = (y_0, y_1, \ldots, y_{k-1})$. The $[\mathbf{n}, \mathbf{k}, \mathbf{d}]$ BCH code corrector is generally operated as the multiplication of a generator matrix M and the input bits $\mathbf{x}$. In other words, the output bits $\mathbf{y}$ of the BCH code corrector are defined as $\mathbf{y}^T = M \cdot \mathbf{x}^T$, and we are able to strictly describe them as

$$\mathbf{y}^T = \begin{bmatrix} c_{n-k} & c_{n-k-1} & \cdots & c_0 & 0 & \cdots & 0 \\ 0 & c_{n-k} & c_{n-k-1} & \cdots & c_0 & \cdots & 0 \\ \vdots & \ddots & & \ddots & & \ddots & \vdots \\ 0 & \cdots & 0 & c_{n-k} & c_{n-k-1} & \cdots & c_0 \end{bmatrix}_{k\times n} \cdot \mathbf{x}^T. \quad (1)$$

Note that $\mathbf{x}^T$ denotes the transpose of $\mathbf{x}$, and the generate matrix M is composed of the coefficient of generator polynomial $G(x) = c_{n-k}x^{n-k} + c_{n-k-1}x^{n-k-1} + \cdots + c_1 x + c_0$ [6, 9].

3.1. Output Distribution of [7, 4, 3] BCH Code Corrector. The $[\mathbf{7}, \mathbf{4}, \mathbf{3}]$ BCH code corrector is generated by a generator polynomial $G(x)$ of $[\mathbf{7}, \mathbf{4}, \mathbf{3}]$ BCH code. Since the generator polynomial of $[\mathbf{7}, \mathbf{4}, \mathbf{3}]$ BCH code is $G(x) = x^3+x+1$, therefore the generator matrix M is generated as

$$M = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}. \quad (2)$$

Let the input bits be $\mathbf{x} = (x_1, x_2, x_3, x_4, x_5, x_6, x_7)$ and let the output bits of the $[\mathbf{7}, \mathbf{4}, \mathbf{3}]$ BCH code corrector be $\mathbf{y} = (y_1, y_2, y_3, y_4)$; then we should represent the output bits as $y_i = x_i \oplus x_{i+2} \oplus x_{i+3}$, where $1 \le i \le 4$. Let X be a random variable on the input bit, and let ϵ be the input bias; we are able to describe the distribution of input bit as $P(X = 0) = 1/2 + \epsilon$ and $P(X = 1) = 1/2 - \epsilon$. Let Z be a random variable on the input bits which are operated by multiplication where the value of matrix entry is 1, and let Y be a random variable on the output bit. The distribution of output bit is represented as $P(Y = 0) = 1/2+4\epsilon^3$ and $P(Y = 1) = 1/2-4\epsilon^3$. Since the input bits are independent by our assumptions, the distribution of output bit of the BCH code corrector is calculated as in Table 2. Table 2 is used to verify the properties of the BCH code corrector as well.

3.2. Mutual Independence of [7, 4, 3] BCH Code Corrector. We examine the $[\mathbf{7}, \mathbf{4}, \mathbf{3}]$ BCH code corrector from the viewpoint of mutual independence. Table 2 is used to verify the dependence of output bits of the BCH code corrector in our inspection. In this subsection, we first describe the definition of mutual independence and verify that the output bits of the BCH code corrector are not mutually independent. Finally, we derive the conjecture that the output bits of the BCH code corrector are not mutually independent, even though the input bits are bitwise independent, due to the intrinsic property of the generator matrix of the BCH code.

Definition 1 (mutual independence [16]). Let the random variables $X_1, X_2, \ldots, X_n$ on $\{0, 1, \ldots, 2^m - 1\}$ have the joint probability density distribution $f(x_1, x_2, \ldots, x_n)$, where $x_1, x_2, \ldots, x_n \in \{0, 1\}^m$, and the marginal probability density distributions $f_1(x_1), f_2(x_2), \ldots, f_n(x_n)$, respectively, for every k $(k = 2, 3, \ldots, n)$ and every subset $\{i_1, i_2, \ldots, i_k\}$ of k distinct values drawn from the set of the first n natural numbers. Then, $X_{i_1}, X_{i_2}, \ldots, X_{i_k}$ are mutually independent if and only if

$$f\left(x_{i_1}, x_{i_2}, \ldots, x_{i_k}\right) = f_{i_1}\left(x_{i_1}\right) \cdot f_{i_2}\left(x_{i_2}\right) \cdots f_{i_k}\left(x_{i_k}\right). \quad (3)$$

Since the output length of $[\mathbf{7}, \mathbf{4}, \mathbf{3}]$ BCH code corrector is 4 bits, we have to validate the independence of two bits, three bits, and four bits, respectively, in order to verify the mutual independence among the output bits.

Proposition 2. *The two output bits of the [7, 4, 3] BCH code corrector are not independent.*

Proof (Proof of Proposition 2). For $1 \le n \le 7$ and $1 \le m \le 4$, let X_n be a random variable on the n-th bit of the input and let Y_m be a random variable on the m-th bit of the output.

Without loss of generality, we are able to describe the two bits for independence analysis as Y_1 and Y_2, due to the fact that the input bits are independent by the assumption. Thus, $P(Y_1 = 0, Y_2 = 0)$ is calculated as

$$\begin{aligned}
P\left(Y_1 = 0, Y_2 = 0\right) &= P\left(X_1 \oplus X_3 \oplus X_4 = 0, X_2 \oplus X_4 \oplus X_5 = 0\right) = \sum_{i=0}^{1} P\left(X_1 \oplus X_3 \oplus X_4 = 0, X_2 \oplus X_4 \oplus X_5 = 0 \mid X_4 = i\right) \cdot P\left(X_4 = i\right) = \sum_{i=0}^{1} P\left(X_1 \oplus X_3 = i, X_2 \oplus X_5 = i \mid X_4 = i\right) \cdot P\left(X_4 = i\right) \\
&= \sum_{i=0}^{1} P\left(X_1 \oplus X_3 = i, X_2 \oplus X_5 = i\right) \cdot P\left(X_4 = i\right) = \sum_{i=0}^{1} P\left(X_1 \oplus X_3 = i\right) \cdot P\left(X_2 \oplus X_5 = i\right) \cdot P\left(X_4 = i\right) = \left(\left(\frac{1}{2} + \epsilon\right)^2 + \left(\frac{1}{2} - \epsilon\right)^2\right)^2 \cdot \left(\frac{1}{2} + \epsilon\right) + \left(2\left(\frac{1}{2} + \epsilon\right)\left(\frac{1}{2} - \epsilon\right)\right)^2 \cdot \left(\frac{1}{2} - \epsilon\right) = 4\epsilon^4 + 4\epsilon^3 + \frac{1}{4} \neq \left(\frac{1}{2} + 4\epsilon^3\right)^2 = P\left(Y_1 = 0\right) \cdot P\left(Y_2 = 0\right).
\end{aligned} \tag{4}$$

Similarly, $P(Y_1 = 0, Y_2 = 1)$, $P(Y_1 = 1, Y_2 = 0)$, and $P(Y_1 = 1, Y_2 = 1)$ are proven as $P(Y_1 = 0, Y_2 = 0)$. The results are as follows:

$$\begin{aligned}
P\left(Y_1 = 0, Y_2 = 1\right) &= \frac{1}{4} - 4\epsilon^4 \neq \frac{1}{4} - 16\epsilon^6 = P\left(Y_1 = 0\right) \cdot P\left(Y_2 = 1\right), \\
P\left(Y_1 = 1, Y_2 = 0\right) &= \frac{1}{4} - 4\epsilon^4 \neq \frac{1}{4} - 16\epsilon^6 = P\left(Y_1 = 1\right) \cdot P\left(Y_2 = 0\right), \\
P\left(Y_1 = 1, Y_2 = 1\right) &= 4\epsilon^4 - 4\epsilon^3 + \frac{1}{4} \neq 16\epsilon^6 - 4\epsilon^3 + \frac{1}{4} = P\left(Y_1 = 1\right) \cdot P\left(Y_2 = 1\right).
\end{aligned} \tag{5}$$

Therefore, the output bits of the [**7**, **4**, **3**] BCH code corrector are not pairwise independent. □

Proposition 3. *The three output bits of the [7, 4, 3] BCH code corrector are not mutually independent.*

Proof (Proof of Proposition 3). Similarly, Proposition 3 is proven by using the same method as Proposition 2. In the proving of the mutual independence of three bits, we classify two cases: Case 1 is the joint distribution of (Y_1, Y_2, Y_3), (Y_1, Y_3, Y_4), and (Y_2, Y_3, Y_4), and Case 2 is the joint distribution of (Y_1, Y_2, Y_4). Due to the characteristic of the generator matrix and generator polynomial, Case 1 shares two bits in the post-processing operation; on the other hand, Case 2 shares only one bit in the post-processing operation. The proofs of Cases 1 and 2 are as follows. □

Case 1 (joint distribution of (Y_1, Y_2, Y_3), (Y_1, Y_3, Y_4) and (Y_2, Y_3, Y_4)). For $1 \leq n \leq 7$ and $1 \leq m \leq 4$, let X_n be a random variable on the n-th bit of the input and let Y_m be a random variable on the m-th bit of the output. Without loss of generality, we are able to represent the three output bits for verifying the mutual independence as Y_1, Y_2, and Y_3, due to the fact that the input bits are bitwise independent by the assumption. Thus, $P(Y_1 = 0, Y_2 = 0, Y_3 = 0)$ is calculated as

$$\begin{aligned}
P\left(Y_1 = 0, Y_2 = 0, Y_3 = 0\right) &= P\left(X_1 \oplus X_3 \oplus X_4 = 0, X_2 \oplus X_4 \oplus X_5 = 0, X_3 \oplus X_5 \oplus X_6 = 0\right) \\
&= \sum_{i=0}^{1} \sum_{j=0}^{1} P\left(X_1 \oplus X_3 = i, X_2 = i \oplus j, X_3 \oplus X_6 = j\right) \cdot P\left(X_4 = i, X_5 = j\right) \\
&= \sum_{i=0}^{1} \sum_{j=0}^{1} P\left(X_1 \oplus X_3 = i, X_2 = i \oplus j, X_3 \oplus X_6 = j\right) \cdot P\left(X_4 = i\right) \cdot P\left(X_5 = j\right) = 6\epsilon^4 + 4\epsilon^3 + \frac{1}{8} \neq 64\epsilon^9 + 24\epsilon^6 + 3\epsilon^3 + \frac{1}{8} = P\left(Y_1 = 0\right) \cdot P\left(Y_2 = 0\right) \cdot P\left(Y_3 = 0\right).
\end{aligned} \tag{6}$$

The similar arguments can be applied to $P(Y_1 = 0, Y_2 = 0, Y_3 = 1)$, $P(Y_1 = 0, Y_2 = 1, Y_3 = 0), \ldots, P(Y_1 = 1, Y_2 = 1, Y_3 = 1)$.

Case 2 (joint distribution of (Y_1, Y_2, Y_4)). For $1 \leq n \leq 7$ and $1 \leq m \leq 4$, let X_n be a random variable on the n-th bit of the input and let Y_m be a random variable on the m-th bit of the output. We can represent the three bits for analysis as Y_1, Y_2, and Y_4, due to the fact that the input bits are independent by the assumption. Thus, $P(Y_1 = 0, Y_2 = 0, Y_4 = 0)$ is calculated as

$$\begin{aligned}
P\left(Y_1 = 0, Y_2 = 0, Y_4 = 0\right) &= P\left(X_1 \oplus X_3 \oplus X_4 = 0, X_2 \oplus X_4 \oplus X_5 = 0, X_4 \oplus X_6 \oplus X_7 = 0\right) \\
&= \sum_{i=0}^{1} P\left(X_1 \oplus X_3 = i, X_2 \oplus X_5 = i, X_6 \oplus X_7 = i\right) \cdot P\left(X_4 = i\right) = \sum_{i=0}^{1} P\left(X_1 \oplus X_3 = i\right) \cdot P\left(X_2 \oplus X_5 = i\right) \cdot P\left(X_6 \oplus X_7 = i\right) \cdot P\left(X_4 = i\right) = 16\epsilon^7 + 6\epsilon^4 + 3\epsilon^3 + \frac{1}{8} \neq 64\epsilon^9 + 24\epsilon^6 + 3\epsilon^3 + \frac{1}{8} = P\left(Y_1 = 0\right) \cdot P\left(Y_2 = 0\right) \cdot P\left(Y_4 = 0\right).
\end{aligned} \tag{7}$$

The similar arguments can be applied to $P(Y_1 = 0, Y_2 = 0, Y_4 = 1)$, $P(Y_1 = 0, Y_2 = 1, Y_4 = 0), \ldots, P(Y_1 = 1, Y_2 = 1, Y_4 = 1)$. Therefore the three output bits of the $[\mathbf{7}, \mathbf{4}, \mathbf{3}]$ BCH code corrector are not mutually independent.

Proposition 4. *The four output bits of the [7, 4, 3] BCH code corrector are not mutually independent.*

Proof (Proof of Proposition 4). For $1 \le n \le 7$ and $1 \le m \le 4$, let X_n be a random variable on the n-th bit of the input and let Y_m be a random variable on the m-th bit of the output. We represent the four bits for analysis as Y_1, Y_2, Y_3, and Y_4. We can prove Proposition 4 using the same method as Proposition 2. The following are the results of the mutual independence of four bits.

$$\begin{aligned} P\left(Y_1 = 0, Y_2 = 0, Y_3 = 0, Y_4 = 0\right) &= P\left(X_1 \oplus X_3 \oplus X_4 = 0, X_2 \oplus X_4 \oplus X_5 = 0, X_3 \oplus X_5 \oplus X_6 = 0, X_4 \oplus X_6 \oplus X_7 = 0\right) \\ &= \sum_{i=0}^{1}\sum_{j=0}^{1}\sum_{k=0}^{1}\sum_{l=0}^{1} P\left(X_1 = i \oplus j, X_2 = j \oplus k, i \oplus k \oplus l = 0, X_7 = j \oplus l\right) \cdot P\left(X_3 = i\right) \\ &\quad \cdot P\left(X_4 = j\right) \cdot P\left(X_5 = k\right) \cdot P\left(X_6 = l\right) = 8\epsilon^7 + 7\epsilon^4 + \frac{7}{2\epsilon^3} + \frac{1}{16} \\ &\neq 256\epsilon^{12} + 128\epsilon^9 + 24\epsilon^6 + 2\epsilon^3 + \frac{1}{16} \\ &= P\left(Y_1 = 0\right) \cdot P\left(Y_2 = 0\right) \cdot P\left(Y_3 = 0\right) \cdot P\left(Y_4 = 0\right). \end{aligned} \tag{8}$$

The similar arguments can be applied to $P(Y_1 = 0, Y_2 = 0, Y_3 = 0, Y_4 = 1)$, $P(Y_1 = 0, Y_2 = 0, Y_3 = 1, Y_4 = 0), \ldots, P(Y_1 = 1, Y_2 = 1, Y_3 = 1, Y_4 = 1)$, and we obtain the fact that the four output bits of the $[\mathbf{7}, \mathbf{4}, \mathbf{3}]$ BCH code corrector are not mutually independent. □

We find out the fact that the output bits of the $[\mathbf{7}, \mathbf{4}, \mathbf{3}]$ BCH code corrector are not mutually independent even though the input bits are bitwise independent. We should also conjecture that the output bits of the BCH code corrector are not mutually independent due to the inherent characteristic of the generator polynomial and generator matrix such that there are sequential degrees in the generator polynomial of BCH code, or the rank of generator matrix. The following are the examples of the generator polynomials of BCH code [9, 17].

(i) The generator polynomial of $[\mathbf{15}, \mathbf{11}, \mathbf{1}]$ BCH code is $G(x) = x^4 + x + 1$.

(ii) The generator polynomial of $[\mathbf{15}, \mathbf{5}, \mathbf{3}]$ BCH code is $G(x) = x^{10} + x^8 + x^5 + x^4 + x^2 + x + 1$.

(iii) The generator polynomial of $[\mathbf{31}, \mathbf{26}, \mathbf{1}]$ BCH code is $G(x) = x^5 + x^2 + 1$.

(iv) The generator polynomial of $[\mathbf{31}, \mathbf{21}, \mathbf{2}]$ BCH code is $G(x) = x^{10} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1$.

(v) The generator polynomial of $[\mathbf{255}, \mathbf{247}, \mathbf{3}]$ BCH code is $G(x) = x^8 + x^4 + x^3 + x^2 + 1$.

(vi) The generator polynomial of $[\mathbf{255}, \mathbf{231}, \mathbf{7}]$ BCH code is $G(x) = x^{24} + x^{23} + x^{21} + x^{20} + x^{19} + x^{17} + x^{16} + x^{15} + x^{13} + x^9 + x^7 + x^6 + x^5 + x^4 + x^3 + x^2 + 1$.

4. Degree of Dependence

In this section, we define a new measure *degree of dependence* which calculates the difference between the distribution of output bits of the BCH code corrector and the distribution where the output bits are independent. In order to formally define the *degree of dependence* of a given distribution, we first define Δ^k which denotes the maximum difference between the k-dimensional distributions of the given distribution and the distributions given by k independent random variables, where the degree of dependence among k bits is assessed.

Definition 5 (degree of dependence). Let $(Y_1, Y_2, \ldots, Y_n)$ be a given random vector on $\{0,1\}^n$ with the joint distribution $\mathscr{D}$. For any $k = 2, 3, \ldots, n$, $\Delta^k(\mathscr{D})$ denotes the maximum difference between the joint distribution $\mathscr{D}$ and the distribution given by k independent random variables. That is, $\Delta^k(\mathscr{D})$ is defined as

$$\begin{aligned} \Delta^k\left(\mathscr{D}\right) = \max_{1 \le i_1 < \cdots < i_k \le n} \Big| & P\left(Y_{i_1} = y_{i_1}, \ldots, Y_{i_k} = y_{i_k}\right) \\ & - P\left(Y_{i_1} = y_{i_1}\right) \cdots P\left(Y_{i_k} = y_{i_k}\right) \Big|, \end{aligned} \tag{9}$$

and then the *degree of dependence* of the given distribution $\mathscr{D}$, $DoD_{\mathscr{D}}$, is denoted by

$$DoD_{\mathscr{D}} = \max\left\{\Delta^2\left(\mathscr{D}\right), \Delta^3\left(\mathscr{D}\right), \ldots, \Delta^n\left(\mathscr{D}\right)\right\}. \tag{10}$$

On the other hand, there is another method measuring the dependence such as *k-wise δ-dependent* [18]. *k-wise δ-dependent* is determined by using the distance between the joint uniform distribution of k random variables and the joint distribution of k random variables. It is defined as follows.

Definition 6 (k-wise δ-dependent). Let $\mathscr{D}$ be a given joint distribution on $\{0,1\}^n$ and let $\mathscr{U}$ be the uniform distribution

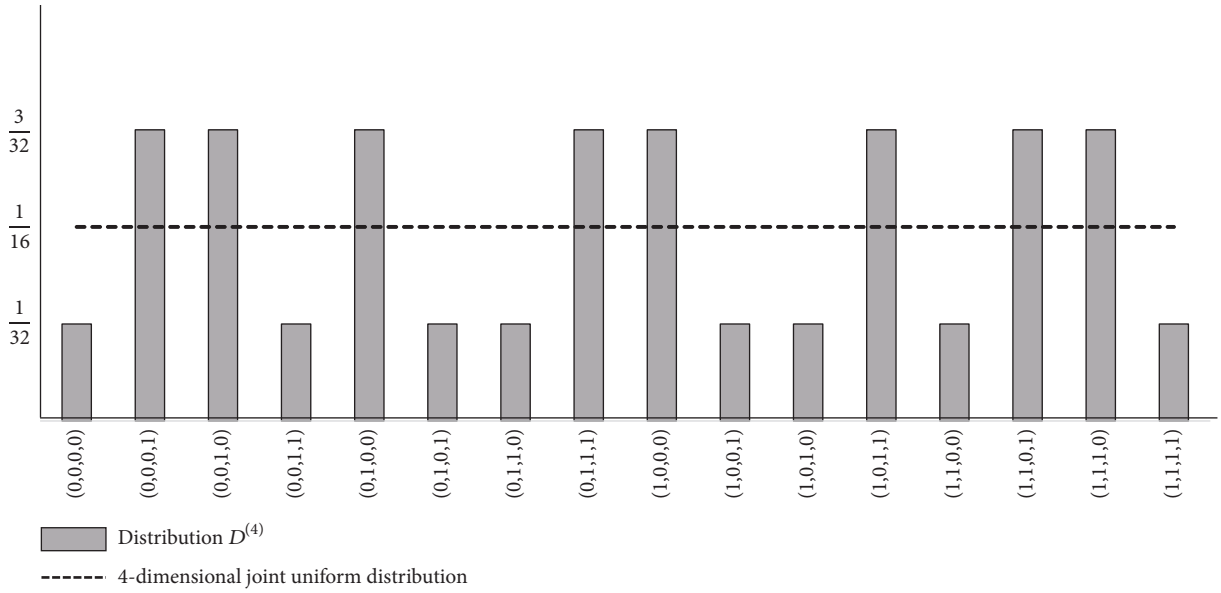

Figure 3: Distribution $\mathscr{D}^{(4)}$ for comparing *degree of dependence* and *k-wise δ-dependence*.

on $\{0,1\}^n$. Then $\mathscr{D}$ is said to be k-wise δ-dependent if for any subset S of the index set $\{1,2,\ldots,n\}$ such that $|S| \leq k \leq n$,

$$\|\mathscr{D}(S) - \mathscr{U}(S)\| \leq \delta, \tag{11}$$

where the variation distance between two distributions $\mathscr{D}_1$ and $\mathscr{D}_2$ defined over the same probability space Ω is denoted by

$$\|\mathscr{D}_1 - \mathscr{D}_2\| = \sum_{\omega \in \Omega} \left| P_{\mathscr{D}_1}(\omega) - P_{\mathscr{D}_2}(\omega) \right|, \tag{12}$$

and $\mathscr{D}(S)$ and $\mathscr{U}(S)$ are the distributions $\mathscr{D}$ and $\mathscr{U}$, respectively, restricted to the subset S.

Degree of dependence is similar to the concept of *k-wise δ-dependence*. By Definition 6, *k-wise δ-dependence* is measured as the sum of the difference between the joint uniform distribution of k random variables and the given k dimensional joint distributions. However, *degree of dependence* measures the maximum difference between the distribution of output bits of the BCH code corrector and the distribution where the output bits are independent. The distribution where the output bits are independent is not a fixed distribution such as the uniform distribution, since it just satisfies (3) in Definition 1. Hence the joint uniform distribution is not equal to the distribution where the output bits are independent. In order to show the difference between *degree of dependence* and *k-wise δ-dependence*, we describe the following example.

Example. For $1 \leq i, m \leq 4$, let Y_i be a random variable on the i-th bit, and let $\mathscr{U}^{(m)}$ be the m-dimensional joint uniform distribution. That is, $P_{\mathscr{U}^{(m)}}(Y_1 = y_1, Y_2 = y_2, \ldots, Y_m = y_m) = (1/2)^m$, for all $y_i \in \{0,1\}$. For example, $P_{\mathscr{U}^{(4)}}(Y_1 = y_1, Y_2 = y_2, Y_3 = y_3, Y_4 = y_4) = 1/16$, for all $y_i \in \{0,1\}$. Let $\mathscr{D}^{(4)}$ be the joint distribution on $\{0,1\}^4$ given by Figure 3.

Then, the marginal distribution from the joint distribution is calculated as

$$\begin{aligned} P(Y_i = y_i) &= \frac{1}{2}, \quad \text{for } 1 \leq i \leq 4, \\ P(Y_i = y_i, Y_j = y_j) &= \frac{1}{4}, \\ &\qquad \text{for } 1 \leq i < j \leq 4, \\ P(Y_i = y_i, Y_j = y_j, Y_k = y_k) &= \frac{1}{8}, \\ &\qquad \text{for } 1 \leq i < j < k \leq 4. \end{aligned} \tag{13}$$

By Definition 5, *degree of dependence* is calculated as $1/32$, since

$$\begin{aligned} \Delta^2\left(\mathscr{D}^{(4)}\right) &= \max_{1 \leq i < j \leq 4} \left| P(Y_i = y_i, Y_j = y_j) - P(Y_i = y_i) \right. \\ &\quad \left. \cdot P(Y_j = y_j) \right| = 0, \\ \Delta^3\left(\mathscr{D}^{(4)}\right) &= \max_{1 \leq i < j < k \leq 4} \left| P(Y_i = y_i, Y_j = y_j, Y_k = y_k) \right. \\ &\quad \left. - P(Y_i = y_i) \cdot P(Y_j = y_j) \cdot P(Y_k = y_k) \right| = 0, \\ \Delta^4\left(\mathscr{D}^{(4)}\right) &= \max \left| P(Y_1 = y_1, Y_2 = y_2, Y_3 = y_3, Y_4 = y_4) \right. \\ &\quad - P(Y_1 = y_1) \cdot P(Y_2 = y_2) \cdot P(Y_3 = y_3) \\ &\quad \left. \cdot P(Y_4 = y_4) \right| = \max \left\{ \left| \frac{1}{32} - \frac{1}{16} \right|, \left| \frac{3}{32} - \frac{1}{16} \right| \right\} \\ &= \frac{1}{32}. \end{aligned} \tag{14}$$

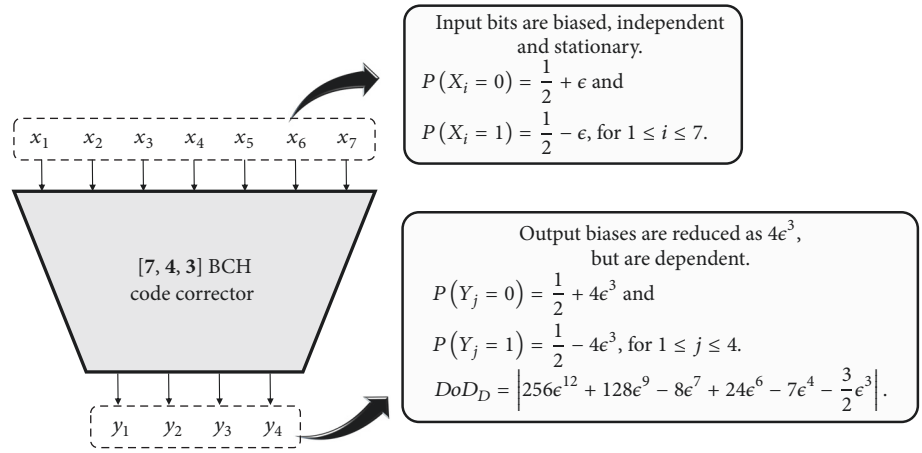

Figure 4: *Degree of dependence* of the [7, 4, 3] BCH code corrector.

Therefore,

$$DoD_{\mathscr{D}} = \max\left\{\Delta^2\left(\mathscr{D}^{(4)}\right), \Delta^3\left(\mathscr{D}^{(4)}\right), \Delta^4\left(\mathscr{D}^{(4)}\right)\right\} = \max\left\{0, 0, \frac{1}{32}\right\} = \frac{1}{32}. \tag{15}$$

On the other hand, by Definition 6, $\mathscr{D}^4$ is 4-wise 1/2-dependent ($\delta = 1/2$) and is calculated as

$$\sum \left|P\left(Y_1 = y_1, Y_2 = y_2, Y_3 = y_3, Y_4 = y_4\right) - P_{\mathscr{U}^{(4)}}\left(Y_1 = y_1, Y_2 = y_2, Y_3 = y_3, Y_4 = y_4\right)\right| = 8 \times \left|\frac{1}{32} - \frac{1}{16}\right| + 8 \times \left|\frac{3}{32} - \frac{1}{16}\right| = \frac{1}{2}. \tag{16}$$

5. Degree of Dependence of the [7, 4, 3] BCH Code Corrector

We verify the fact that the output bits of the $[\mathbf{7}, \mathbf{4}, \mathbf{3}]$ BCH code corrector are not mutually independent in Section 3 and formally define *degree of dependence* of a given distribution in Section 4. In this section, we exploit *degree of dependence* in order to measure the difference between the distribution of output bits of the $[\mathbf{7}, \mathbf{4}, \mathbf{3}]$ BCH code corrector and the distribution given by four independent random variables. In order to examine *degree of dependence* of the $[\mathbf{7}, \mathbf{4}, \mathbf{3}]$ BCH code corrector, we assume that the input bits are bitwise independent, biased, and stationary as in Section 3. Figure 4 shows the assumption of our inspection and *degree of dependence* of the $[\mathbf{7}, \mathbf{4}, \mathbf{3}]$ BCH code corrector.

For $1 \le i \le 7$ and $1 \le j \le 4$, let the input bits be $\mathbf{x} = (x_1, x_2, x_3, x_4, x_5, x_6, x_7)$, for all $x_i \in \{0, 1\}$, and let ϵ be the bias of input bits where the range of input bias is $0 < \epsilon < 1/2$. Then the distribution of input bits is represented as

$$P\left(X_i = 0\right) = \frac{1}{2} + \epsilon \quad \text{and} \quad P\left(X_i = 1\right) = \frac{1}{2} - \epsilon. \tag{17}$$

Let $\mathscr{D}$ be the distribution of output bits of the $[\mathbf{7}, \mathbf{4}, \mathbf{3}]$ BCH code corrector, and let the output bits be $\mathbf{y} = (y_1, y_2, y_3, y_4)$ for all $y_j \in \{0, 1\}$. From Table 2, the output distribution of the $[\mathbf{7}, \mathbf{4}, \mathbf{3}]$ BCH code corrector is described as

$$P\left(Y_j = 0\right) = \frac{1}{2} + 4\epsilon^3 \quad \text{and} \quad P\left(Y_j = 1\right) = \frac{1}{2} - 4\epsilon^3. \tag{18}$$

In order to examine *degree of dependence* of the $[\mathbf{7}, \mathbf{4}, \mathbf{3}]$ BCH code corrector, we employ the results which are the output distribution of the $[\mathbf{7}, \mathbf{4}, \mathbf{3}]$ BCH code corrector in Table 2, Propositions 2, 3, and 4, and our assumption. For $2 \le j \le 4$, $\Delta^j(\mathscr{D})$ is calculated as the maximum difference between the j-dimensional distribution of output bits of the $[\mathbf{7}, \mathbf{4}, \mathbf{3}]$ BCH code corrector and the distribution where the j bits among the outputs are independent. We obtain $\Delta^2(\mathscr{D})$, $\Delta^3(\mathscr{D})$, and $\Delta^4(\mathscr{D})$ by using (9) in Definition 5, and they can be represented as functions on ϵ for $0 < \epsilon < 1/2$:

$$\Delta^2\left(\mathscr{D}\right) = \left|16\epsilon^6 - 4\epsilon^4\right|, \quad \Delta^3\left(\mathscr{D}\right) = \left|64\epsilon^9 + 24\epsilon^6 - 6\epsilon^4 - \epsilon^3\right|, \quad \Delta^4\left(\mathscr{D}\right) = \left|256\epsilon^{12} + 128\epsilon^9 - 8\epsilon^7 + 24\epsilon^6 - 7\epsilon^4 - \frac{3}{2\epsilon^3}\right|. \tag{19}$$

Based on these results, Figure 5 is depicted as the graphs of $\Delta^2(\mathscr{D})$, $\Delta^3(\mathscr{D})$, and $\Delta^4(\mathscr{D})$ for $0 < \epsilon < 1/2$. Since $DoD_{\mathscr{D}}$ is determined by (10), the maximum value of $\Delta^2(\mathscr{D})$, $\Delta^3(\mathscr{D})$, and $\Delta^4(\mathscr{D})$, we can derive $DoD_{\mathscr{D}}$ of the $[\mathbf{7}, \mathbf{4}, \mathbf{3}]$ BCH code corrector as

$$DoD_{\mathscr{D}} = \left|256\epsilon^{12} + 128\epsilon^9 - 8\epsilon^7 + 24\epsilon^6 - 7\epsilon^4 - \frac{3}{2\epsilon^3}\right| \quad \text{for } 0 < \epsilon < \frac{1}{2}. \tag{20}$$

When ϵ is $0.25, 0.125, 0.1$, and 0.01, $\Delta^2(\mathscr{D})$, $\Delta^3(\mathscr{D})$, $\Delta^4(\mathscr{D})$, and $DoD_{\mathscr{D}}$ are described in Table 3. From Figure 5 and

Table 3: Theoretical results for $\epsilon = 0.25, 0.125$, and 0.1.

	$\Delta^2(\mathscr{D})$	$\Delta^3(\mathscr{D})$	$\Delta^4(\mathscr{D})$	$DoD_{\mathscr{D}}$
$\epsilon = 0.25$	1.17×10^{-2}	3.30×10^{-2}	4.49×10^{-2}	4.49×10^{-2}
$\epsilon = 0.125$	9.16×10^{-4}	3.33×10^{-3}	4.55×10^{-3}	4.55×10^{-3}
$\epsilon = 0.1$	3.84×10^{-4}	1.58×10^{-3}	2.18×10^{-3}	2.18×10^{-3}

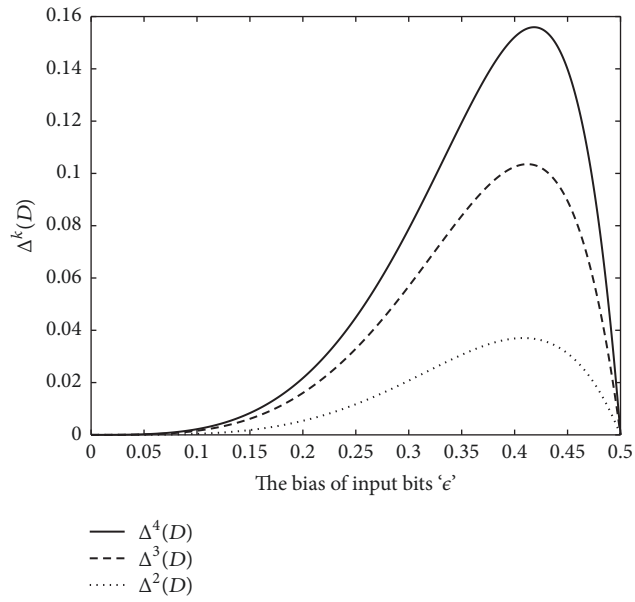

Figure 5: The graph of Δ^2, Δ^3, and Δ^4 depending on ϵ.

Table 3, we should conjecture that $DoD_{\mathscr{D}}$ of the [**7**, **4**, **3**] BCH code corrector is determined by $\Delta^4(\mathscr{D})$ and also derive the fact that $DoD_{\mathscr{D}}$ is close to 0, when the bias ϵ of input bits is close to 0.

6. Experimental Results and Applications

6.1. Experimental Results. We have conducted experiments to support the theoretical results and examine *degree of dependence* depending on the bias of input bits. In order to verify our examination, we have obtained experimental data sequences from Numpy [19]. Numpy is the library of Python and provides various probability distributions such as Binomial distribution, Normal distribution, and Poisson distribution. We have collected the experimental data from the Bernoulli distribution with success probability p of each trial by using Numpy, in order to make experimental data satisfying our assumptions that the input bits are biased, independent, and stationary. In this experiment, we have set that the success probability p of the Bernoulli distribution is 0.25, 0.375, and 0.4, and then we have collected experimental data; data size is 30 MB, 300 MB, and 3 GB, respectively. Note that $p = 0.25$ denotes that the bias of input bits is 0.25, $p = 0.375$ denotes that the bias of input bits is 0.125, and $p = 0.4$ denotes that the bias of input bits is 0.1.

Table 4 shows the experimental results for $\Delta^2(\mathscr{D})$, $\Delta^3(\mathscr{D})$, $\Delta^4(\mathscr{D})$, and $DoD_{\mathscr{D}}$ where the input bias ϵ is 0.25, 0.125, and 0.1, and the experimental data size is 30 MB, and Tables 5 and 6 show the results of the experimental data of size 300 MB and 3 GB, respectively. The reason for performing the experiments for different data sizes is to accurately inspect our theoretical result. Since the difference values between the theoretical result and experimental results in Tables 4, 5, and 6 are less than $7.83 \cdot 10^{-4}$, $1.52 \cdot 10^{-4}$, and $1.10 \cdot 10^{-4}$, respectively, we are able to state that *degree of dependence* in Tables 4, 5, and 6 is similar to the theoretical results in Table 3. Specially, as the data size increases, and the accuracy of the experiment increases; thus, the values in Table 6 are more approximate to the theoretical results. Therefore the experimental results support our theoretical result for *degree of dependence*.

6.2. Application of the [7, 4, 3] BCH Code Corrector in TRNG. In order to harvest the entropy from the noise source, RNGs in the lightweight environment usually collect sensor-based noise sources such as microphone, accelerometer, magnetometer, gyroscope, temperature, and humidity [20, 21]. However, since these noise sources have low entropy [20, 21], RNGs have to apply post-processing component so as to reduce biases and to increase the entropy per bit.

It is possible to consider various post-processing components which are appropriate in the lightweight environment, except for NIST conditioning components where the heavy cryptographic algorithm is involved. Although the von Neumann corrector is lightweight, it is also not suitable from the security point of view due to some drawbacks such as adversary bias; on the contrary, the linear code correctors

Table 4: Experimental results for 30 MB data where ϵ is 0.25, 0.125, and 0.1.

	$\Delta^2(\mathscr{D})$	$\Delta^3(\mathscr{D})$	$\Delta^4(\mathscr{D})$	$DoD_{\mathscr{D}}$
$\epsilon = 0.25$	1.25×10^{-2}	3.31×10^{-2}	4.48×10^{-2}	4.48×10^{-2}
$\epsilon = 0.125$	7.19×10^{-4}	3.16×10^{-3}	4.70×10^{-3}	4.70×10^{-3}
$\epsilon = 0.1$	5.95×10^{-4}	1.42×10^{-3}	2.20×10^{-3}	2.20×10^{-3}

Table 5: Experimental results for 300 MB data where ϵ is 0.25, 0.125, and 0.1.

	$\Delta^2(\mathscr{D})$	$\Delta^3(\mathscr{D})$	$\Delta^4(\mathscr{D})$	$DoD_{\mathscr{D}}$
$\epsilon = 0.25$	1.19×10^{-2}	3.30×10^{-2}	4.50×10^{-2}	4.50×10^{-2}
$\epsilon = 0.125$	9.87×10^{-4}	3.33×10^{-3}	4.58×10^{-3}	4.58×10^{-3}
$\epsilon = 0.1$	5.23×10^{-4}	1.71×10^{-3}	2.18×10^{-3}	2.18×10^{-3}

Table 6: Experimental results for 3 GB data where ϵ is 0.25, 0.125, and 0.1.

	$\Delta^2(\mathscr{D})$	$\Delta^3(\mathscr{D})$	$\Delta^4(\mathscr{D})$	$DoD_{\mathscr{D}}$
$\epsilon = 0.25$	1.18×10^{-2}	3.30×10^{-2}	4.50×10^{-2}	4.50×10^{-2}
$\epsilon = 0.125$	9.66×10^{-4}	3.38×10^{-3}	4.56×10^{-3}	4.56×10^{-3}
$\epsilon = 0.1$	4.50×10^{-4}	1.64×10^{-3}	2.23×10^{-3}	2.23×10^{-3}

perform much better than von Neumann corrector [9]. The [**7**, **4**, **3**] BCH code corrector is applicable to resource constrained environments, such as IoT, embedded, and mobile devices [22, 23], since the code size and the circuit complexity of the proposed code corrector are small as compared with other code correctors and NIST's conditioning components.

It is also possible to iteratively use the [**7**, **4**, **3**] BCH code corrector, when the size of the collected noise source is bigger than the input size of the BCH code corrector. For instance, if the size of the collected noise source is 32 bits, the [**7**, **4**, **3**] BCH code corrector is four times iteratively processed with separated 7 input bits, and the residual 4 input bits are discarded or stored for the next generation. From our result which finds out the relation between the bias of input bits and *degree of dependence* among output bits, the [**7**, **4**, **3**] BCH code corrector is controllable to *degree of dependence* corresponding to the bias of input bits. The entropy sources are able to be effectively managed by the [**7**, **4**, **3**] BCH code corrector according to the environment of collecting the noise sources.

7. Conclusion

We have proposed a new lightweight BCH code corrector that the bitwise dependence of the output bits is controllable. We have focused on the dependence of the output bits and define a new measure *degree of dependence* for making the lightweight BCH code corrector with adjustable dependence corresponding to the bias of input bits. Note that most code correctors have been studied on the subject of reducing the bias of the output bits where the input bits are independent and biased. We have examined the [**7**, **4**, **3**] BCH code corrector from the viewpoint of independence in order to utilize it in the lightweight environment. We have obtained the relation between *degree of dependence* and the bias of input bits through theoretical results and some simulations. Due to its simple structure with small code size and low circuit complexity, the proposed code corrector is able to be applied to lightweight environments such as IoT, embedded, and mobile devices. Moreover the proposed code corrector has the properties of measurable and controllable dependence among output bits as well as reducing the bias of input bits. We expect that the proposed BCH code corrector is utilized to efficiently manage the entropy source in the lightweight environment. In future works, we are planning to study on measuring the *degree of dependence* of various probability distributions from other post-processing components, and we will also study how to apply *degree of dependence* for evaluation of some cryptographically secure random bit sequences.

Data Availability

No data were used to support our study.

Conflicts of Interest

The authors declare that they have no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by Institute for Information & Communications Technology Promotion (IITP) grant funded by the Korean Government (MSIT) (NO.2014-6-00908, Research on the Security of Random Number Generators and Embedded Devices).

References

[1] ISO/IEC 18031, Information technology - Security techniques - Random bit generation, 2011.

[2] W. Killmann and W. Schindler, "A proposal for: Functionality classes and evlauation methodology for true (physical) random number generators," BSI AIS.31, 2001.

[3] E. Barker and J. Kelsey, *Recommendation for Random Bit Generator(RBG) Construction*, NIST Special Publication 800-90C, 2016.

[4] M. S. Turan, E. Barker, J. Kelsey, K. A. McKay, M. L. Baish, and M. Boyle, "Recommendation for the entropy sources used for random bit generation," National Institute of Standards and Technology NIST SP 800-90B, 2018.

[5] A. Vassilev and T. A. Hall, "The importance of entropy to information security," *The Computer Journal*, vol. 47, no. 2, pp. 78–81, 2014.

[6] Y.-S. Kim, J.-W. Jang, and D.-W. Lim, "Linear corrector overcoming minimum distance limitation for secure TRNG from (17, 9, 5) quadratic residue code," *ETRI Journal*, vol. 32, no. 1, pp. 93–101, 2010.

[7] J. von Neumann, "Various techniques used in connection with random digits," *Applied Math Series*, pp. 36–38, 1951.

[8] R. B. Davies, *Exclusive OR (XOR) and Hardware Random Number Generators*, 2002, http://www.robertnz.net/pdf/xor2.pdf.

[9] S. Kwok, Y. Ee, G. Chew, K. Zheng, K. Khoo, and C. Tan, "A Comparison of Post-Processing Techniques for Biased Random Number Generators," in *Information Security Theory and Practice. Security and Privacy of Mobile Devices in Wireless Communication*, vol. 6633 of *Lecture Notes in Computer Science*, pp. 175–190, Springer, Berlin, Germany, 2011.

[10] S. Markovski, D. Gligoroski, and L. Kocarev, "Unbiased Random Sequences from Quasigroup String Transformation," in *Proceedings of the International Workshop on Fast Software Encryption*, pp. 163–180, Springer, Berlin, Germany.

[11] M. Dicht, "Bad and Good Ways of Post-processing Biased Physical Random Numbers," in *Proceedings of the International Workshop on Fast Software Encryption*, pp. 45–62, Springer, Berlin, Germany, 2007.

[12] P. Lacharme, "Post-processing functions for a biased physical random number generator," in *Proceedings of the International Workshop on Fast Software Encryption*, pp. 334–342, Springer, Berlin, Germany, 2008.

[13] Federal Information Processing Standard 180-4, Secure Hash Standard (SHS), 2015.

[14] Federal Information Processing Standard 202, SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions, 2015.

[15] Federal Information Processing Standard 198-1, The Keyed-Hash Message Authentication Code (HMAC), 2008.

[16] R. V. Hogg and A. T. Craig, *Introduction to Mathematical Statistics*, Macmillan, 4th edition, 1978.

[17] R. Bose, *Information Theory, Coding and Cryptography*, McGrawHillEducation, 3rd edition, 2016.

[18] J. Naor and M. Naor, "Small-bias probability spaces: efficient constructions and applications," *SIAM Journal on Computing*, vol. 22, no. 4, pp. 838–856, 1993.

[19] NumPy, http://www.numpy.org.

[20] C. Hennebert, H. Hossayni, and C. Lauradoux, "Entropy harvesting from physical sensors," in *Proceedings of the 6th ACM Conference on Security and Privacy in Wireless and Mobile Networks, WiSec 2013*, pp. 149–154, ACM, Hungary, April 2013.

[21] K. Wallace, K. Moran, E. Novak, G. Zhou, and K. Sun, "Toward sensor-based random number generation for mobile and IoT devices," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 1189–1201, 2016.

[22] G. C. C. F. Pereira, R. C. A. Alves, F. L. da Silva, R. M. Azevedo, B. C. Albertini, and C. B. Margi, "Performance evaluation of cryptographic algorithms over IoT platforms and operating systems," *Security and Communication Networks*, vol. 2017, Article ID 2046735, 16 pages, 2017.

[23] M. Schramm, R. Dojen, and M. Heigl, "A vendor-neutral unified core for cryptographic operations in GF(p) and GF(2m) based on montgomery arithmetic," *Security and Communication Networks*, vol. 2018, Article ID 4983404, 2018.