entropy

MDPI

Article

Practical Entropy Accumulation for Random Number Generators with Image Sensor-Based Quantum Noise Sources

Youngrak Choi [1], Yongjin Yeom [2] and Ju-Sung Kang [2,*]

[1] Department of Financial Information Security, Kookmin University, Seoul 02707, Republic of Korea; alpha1996@naver.com
[2] Department of Mathematics, Kookmin University, Seoul 02707, Republic of Korea; salt@kookmin.ac.kr
* Correspondence: jskang@kookmin.ac.kr

Abstract: The efficient generation of high-quality random numbers is essential in the operation of cryptographic modules. The quality of a random number generator is evaluated by the min-entropy of its entropy source. The typical method used to achieve high min-entropy of the output sequence is an entropy accumulation based on a hash function. This is grounded in the famous Leftover Hash Lemma, which guarantees a lower bound on the min-entropy of the output sequence. However, the hash function-based entropy accumulation has slow speed in general. For a practical perspective, we need a new efficient entropy accumulation with the theoretical background for the min-entropy of the output sequence. In this work, we obtain the theoretical bound for the min-entropy of the output random sequence through the very efficient entropy accumulation using only bitwise XOR operations, where the input sequences from the entropy source are independent. Moreover, we examine our theoretical results by applying them to the quantum random number generator that uses dark shot noise arising from image sensor pixels as its entropy source.

Keywords: entropy accumulation; random number generator; quantum random noises

Citation: Choi, Y.; Yeom, Y.; Kang, J.-S. Practical Entropy Accumulation for Random Number Generators with Image Sensor-Based Quantum Noise Sources. *Entropy* **2023**, *25*, 1056. https://doi.org/10.3390/e25071056

Academic Editors: Hua-Lei Yin, Kaizhi Huang and Guan-Jie Fan-Yuan

Received: 14 June 2023
Revised: 4 July 2023
Accepted: 11 July 2023
Published: 13 July 2023

1. Introduction

A random number generator (RNG) is an important component of cryptographic systems used by cryptographic modules to generate random numbers. Random numbers are used for various purposes including the generation of cryptographic keys, and their significance has been increasingly highlighted, particularly with the recent emergence of quantum key distribution [1].

An RNG can be divided into three main processes: digitization, entropy accumulation, and pseudo random number generation (PRNG). Digitization is the process of converting entropy sources into binary data. We call the converted binary data the "input sequence". Typically, the input sequence has a low min-entropy. Entropy accumulation is the process of transforming input sequences into data with high min-entropy. We denote the input sequence that has undergone the entropy accumulation process as the "output sequence". PRNG is composed of deterministic algorithms, such as block ciphers or hash functions, and it assumes the output sequence as input and then outputs the final "random number". The operation of the RNG is illustrated in Figure 1.

Although unpredictable random numbers can be generated using a digitized entropy source, this is impractical in cryptographic systems because of the significant amount of time required. The PRNG was used to address this limitation. A PRNG produces the same output with the same input. This implies that the generated random numbers are not unpredictable. However, the PRNG can generate multiple random numbers in a short time because the length of the output is longer than the length of the input. Therefore, if the length of the input of the PRNG is small but random, the output of the PRNG provides good random numbers. Consequently, the high min-entropy of the input sequence can be observed as an important factor in constructing an RNG.

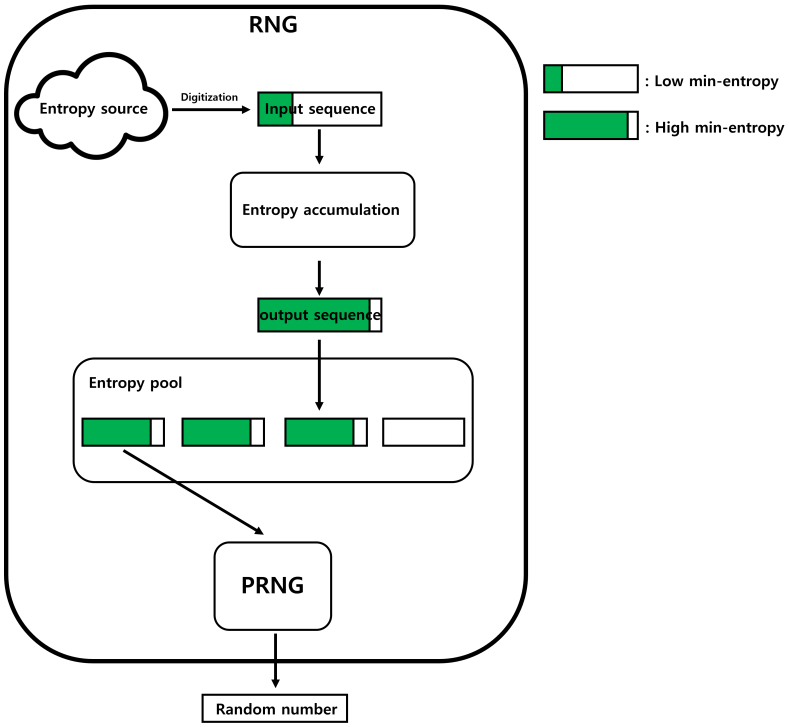


Figure 1. Operation of an RNG.

In the process of entropy accumulation, an accumulation function H is a transformation from an input sequence to an output sequence. Let X be an input sequence and X' be the corresponding output sequence, then the entropy accumulation can be expressed as $X' = H(X)$.

On the other hand, Dodis et al. [2] have proposed that the entropy accumulation is divided into two types depending on the characteristics of the accumulation function H. The first type is called "Slow-Refresh", which is characterized by a high computational complexity of H leading to slower accumulation speed but results in a long output sequence. The second type, so called "Fast-Refresh", features a lower computational complexity of H leading to faster accumulation speed but results in a short output sequence.

Traditional hash function-based entropy accumulation is categorized as Slow-Refresh due to its relatively slow accumulation speed. However, this method is widely utilized because of the theoretical foundation provided by the Leftover Hash Lemma [3]. The Leftover Hash Lemma ensures the lower bound of min-entropy for the output sequence, where a low min-entropy input sequence passes through a hash function H randomly selected from a universal hash family.

There are two major considerations in the Slow-Refresh process, which are the construction of a universal hash family and the random selection of a hash function from the uniformly distributed universal hash family. However, constructing a universal hash family is not a trivial task. One example satisfying an appropriate property is the family of the Hankel matrix [4]. If a Hankel matrix is randomly selected from the uniformly distributed family, and an n-bit input sequence is processed through this selected matrix, then the resulting m-bit output sequence will have a sufficient min-entropy. Similarly, a universal hash family can also be constructed by using Toeplitz matrices [5,6].

In such a matrix-based universal hash family, the matrices typically used have input bit-lengths larger than the output bit-lengths, and the ratio m/n approaches 1 as n increases. Thus, the size of the matrix must be sufficiently large in order to minimize the lost bits. However, this leads to a high computational complexity, that is, this method falls under the Slow-Refresh category, and Slow-Refresh may not always be suitable in practical situations that require rapid entropy accumulation. Therefore, in this work, we focus on Fast-Refresh, which is clearly described in the next subsection.

1.1. Related Works

One of the typical examples of Fast-Refresh without a hash function is Microsoft Windows RNG [7]. Windows RNG uses only the bitwise XOR operation and bit permutation $rot_{(\alpha,n)}$ for the entropy accumulation. In particular, if we employ the following notation, the entropy accumulation operation of Windows RNG can be depicted as shown in Figure 2.

- $Y_1, Y_2, \cdots, Y_l$: n-bit input sequences.
- $\pi : \{0,1,\cdots,n-1\} \to \{0,1,\cdots,n-1\}$, π is one-to-one.
- $A_\pi : \{0,1\}^n \to \{0,1\}^n$, $A_\pi(b_0, b_1, \cdots, b_{n-1}) := A_\pi(b_{\pi(0)}, b_{\pi(1)}, \cdots, b_{\pi(n-1)})$.
- $rot_{(\alpha,n)} : \{0,1,\cdots,n-1\} \to \{0,1,\cdots,n-1\}$, $rot_{(\alpha,n)}(i) := i - \alpha (\mathrm{mod}\ \ n)$.

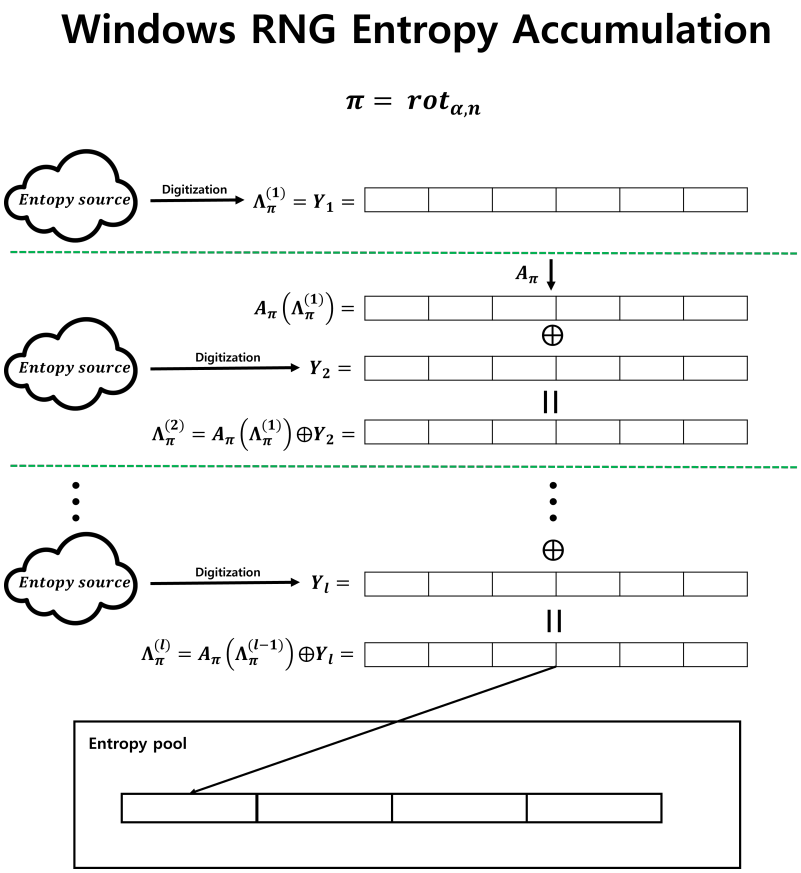


Figure 2. Windows RNG entropy accumulation.

Figure 3 is an example of the $rot_{(3,8)}$ operation on an 8-bit input sequence.

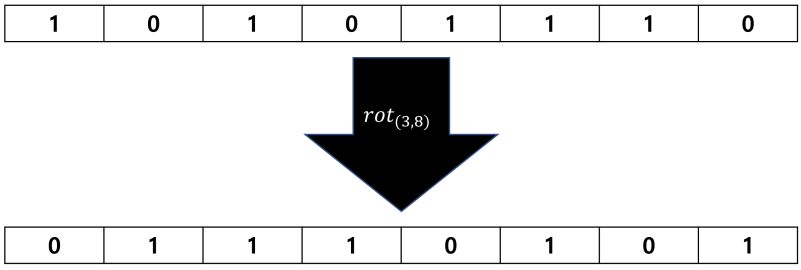


Figure 3. Example of the *rot* permutation when $n = 8$ and $\alpha = 3$.

- $\Lambda_\pi^{(l)} : \{0,1\}^n \to \{0,1\}^n$, $\Lambda_\pi^{(l)} := \begin{cases} Y_1 & \text{if } l = 1 \\ A_\pi(\Lambda_\pi^{(l-1)}) \oplus Y_l & \text{if } l \geq 2 \end{cases}$

Windows RNG accumulates output sequences relatively quickly in an entropy pool because it uses bit permutations and bitwise XOR operations without employing hash functions. Despite these advantages, it has not been proven whether the entropy accumulation of Windows RNG guarantees a lower bound for the min-entropy of the output sequence, as does the Leftover Hash Lemma when using a hash function. Therefore, it has been challenging to consider this method of secure entropy accumulation. However, recent research presented at Crypto 2021 analyzed Microsoft Windows RNG. In [2], the security of Windows RNG was analyzed by providing the number of iterations of bit permutation and bitwise XOR to surpass an arbitrary min-entropy under three conditions. First, the input sequences must be independent. Second, the probability distribution of input sequences must follow the "2-monotone distribution". Third, the "covering number" of the bit permutation must be finite. In [2], it was claimed that the three conditions just mentioned are easy to satisfy. However, satisfying these conditions may be challenging for hardware entropy sources rather than software entropy sources, particularly when managing multiple entropy sources. The first and third conditions are easily satisfied, as in the case of Windows RNG. The second condition may seem easy to achieve, but it is challenging. Therefore, to handle entropy sources other than Windows, a more relaxed condition is required than that presented in [2].

1.2. Our Contributions

The contributions of this paper can be summarized into three main aspects. First, we provide Fast-Refresh that does not require hash functions and uses only bitwise XOR operations to generate output sequences. In particular, if we employ the following notation, the proposed entropy accumulation can be depicted as shown in Figure 4.

- $Y_1, Y_2, \cdots, Y_l$: n-bit input sequences.
- $\Gamma^{(l)} = \sum_{j=1}^{l} Y_j$.

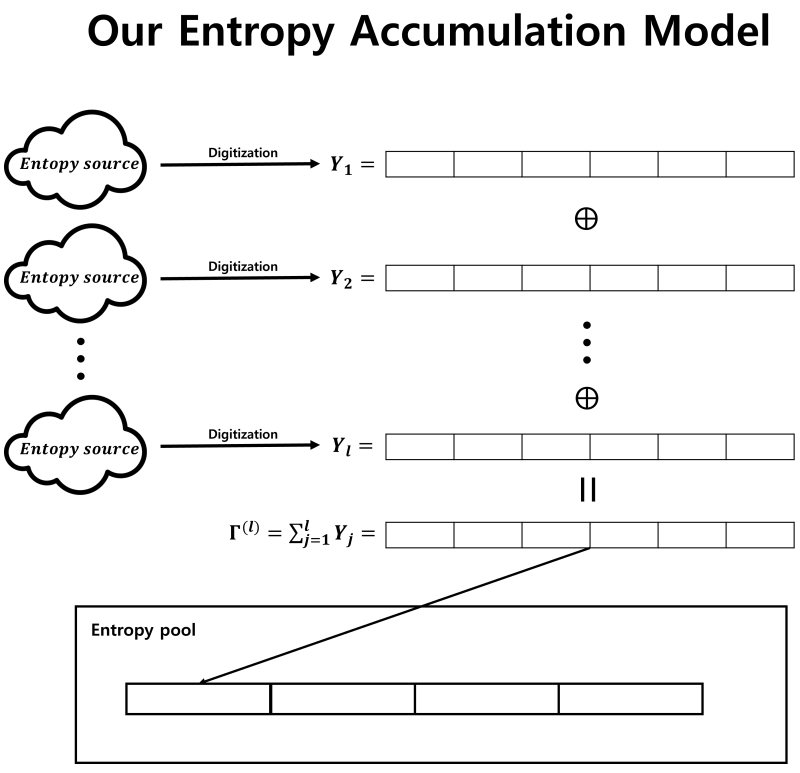


Figure 4. Entropy accumulation using only bitwise XOR operations.

This method requires only two conditions for the input sequences, making it relatively easier to satisfy than Windows RNG entropy accumulation.

Second, we establish a min-entropy lower bound for a secure random number generator and demonstrate that our entropy accumulation successfully surpasses this lower bound when applied to our RNG. We used quantum random number generators (QRNGs) as an entropy source. QRNGs utilize several quantum phenomena to generate high-quality random numbers [8–10].

The first published QRNG was based on radioactive decay. This emerged as the need for random number generators increased alongside the rise of computer simulations in the late 20th century. QRNGs with radioactive decay utilize the random behavior of particles emitted from radioactive materials. This method is still in use today, and, along with methods using the photon, it is one of the most common QRNG [8]. There are various quantum phenomena used for quantum random number generation. For example, the study in [9] uses the interference of photons to generate random numbers. Another example from [11] involves the use of tunneling signals in silicon diodes for random number generation. Furthermore, the work in [12] employs short laser pulses with quantum random phases to generate random numbers.

In this paper, we use the image sensor-based QRNG of [13]. The image sensor-based QRNG generates random numbers using dark shot noise. Dark shot noise is a fluctuation of small current that flows through the pixels of the image sensor even when they do not receive light. It is known that the number of electrons follows the Poisson distribution [14,15]. For this reason, we utilize optical black pixels (OBP) of the image sensor, which do not receive light, as the entropy source. Furthermore, because each pixel outputs dark shot noise independently, all the entropy sources can be considered independent of each other.

Third, we conduct a comparative analysis of our proposed entropy accumulation with other entropy accumulations. The examples chosen for comparison are the Slow-Refresh used in IDQ QRNG and the Fast-Refresh of Windows RNG. When comparing with the Slow-Refresh, we focus on the theoretical differences in the accumulation mechanisms between ours and the IDQ QRNG. Under the view point of Fast-Refresh, we compare efficiency of ours with Windows RNG by evaluating the iteration number of operations.

The remainder of this paper is organized as follows. In Section 2 , we describe the theoretical background and propose our main theorem, which guarantees the lower bound of min-entropy of the output sequence. In Section 3, we describe the process of applying the theory outlined in Section 2 to an image sensor-based QRNG. We establish a min-entropy lower bound based on three standards and provide experimental results demonstrating that the output sequences generated by applying our theory to the input sequences have a min-entropy higher than the established lower bound. In Section 4, we compare our entropy accumulation with other entropy accumulations. The first comparison is with Slow-Refresh of IDQ QRNG. We describe the theoretical background of Slow-Refresh and the Leftover Hash Lemma and explain the operation of IDQ QRNG. Then, we present two limitations of IDQ QRNG and describe the differences between IDQ QRNG and our entropy accumulation. The second comparison is with Fast-Refresh of Windows RNG. We compare the iteration number l, which is obtained when applying each entropy accumulation. The iteration number of Windows RNG is calculated using the theory in [2]. Note that without some additional components, the theory in [2] cannot be directly applied. Section 5 is the conclusion.

2. Theoretical Background and Main Theorem

In this section, we describe the theoretical background of entropy accumulation using only the XOR operation. In particular, we use the following notation:

- $\mathbb{Z}_m^n$: Direct product of n copies of the group $\mathbb{Z}_m$. Note that the bitwise XOR operation corresponds to the $+$ operation over $\mathbb{Z}_2^n$.
- $\mathcal{F}(\mathbb{Z}_m^n)$: the space of all complex valued functions on $\mathbb{Z}_m^n$.
- $\Gamma^{(l)} = \sum_{j=1}^{l} Y_j$, where $Y_j \in \mathbb{Z}_2^n$ is the n-bit random variable that represents the input sequence.

- $\|f\|_{min} = \min\{|f(\mathbf{x})| : \mathbf{x} \in \mathbb{Z}_m^n\}$, $f \in \mathcal{F}(\mathbb{Z}_m^n)$.
- $\|f\|_\infty = \max\{|f(\mathbf{x})| : \mathbf{x} \in \mathbb{Z}_m^n\}$, $f \in \mathcal{F}(\mathbb{Z}_m^n)$.
- $\|f\|_1 = \sum_{\mathbf{x}\in\mathbb{Z}_m^n} |f(\mathbf{x})|$, $f \in \mathcal{F}(\mathbb{Z}_m^n)$.
- D_X : probability distribution of the random variable X.
- $H_{min}(D_X) = -\log_2 \|D_X\|_\infty$. $H_{min}(D_X)$ implies the min-entropy of D_X.

We show that as the number of input sequences required to generate one output sequence, represented by l, approaches infinity, $H_{min}(D_{\Gamma^{(l)}})$ converges to n. Furthermore, we provide the optimal value of l necessary to surpass the specified min-entropy $\alpha(< n)$. First, we provide a solution for the case $n = 1$ and explain why this solution is inappropriate for the general n-bit case. Thereafter, we provide a general solution using a Discrete Fourier Transform and Convolution.

First, we show why the problem we are trying to solve is challenging. The difficult point of our problem is that in order to determine the value of $D_{\Gamma^{(l)}}$, complex linear operations must be performed on the function values of D_{Y_l}. For example, suppose $n = 2$, Y_1, Y_2, Y_3 are independent and $D_{Y_1}, D_{Y_2}, D_{Y_3}$ are identical to the distribution D. The distribution D is determined as $D(0,0) = \frac{1}{8}$, $D(0,1) = \frac{1}{4}$, $D(1,0) = \frac{3}{8}$, $D(1,1) = \frac{1}{4}$. Let us calculate $D_{\Gamma^{(3)}}$, which suffices to show the complexity of computation.

$$D_{\Gamma^{(3)}}(0,0) = \sum_{\mathbf{x}\oplus\mathbf{y}\oplus\mathbf{z}=(0,0)} D_{Y_1}(\mathbf{x})D_{Y_2}(\mathbf{y})D_{Y_3}(\mathbf{z}) = D(0,0)D(0,0)D(0,0)+$$
$$D(0,0)D(0,1)D(0,1) + \cdots + D(1,1)D(1,1)D(0,0) = \frac{124}{512} \approx 0.242.$$

$$D_{\Gamma^{(3)}}(0,1) = \sum_{\mathbf{x}\oplus\mathbf{y}\oplus\mathbf{z}=(0,1)} D_{Y_1}(\mathbf{x})D_{Y_2}(\mathbf{y})D_{Y_3}(\mathbf{z}) = D(0,0)D(0,0)D(0,1)+$$
$$D(0,0)D(0,1)D(0,0) + \cdots + D(1,1)D(1,1)D(0,1) = \frac{128}{512} = 0.25.$$

$$D_{\Gamma^{(3)}}(1,0) = \sum_{\mathbf{x}\oplus\mathbf{y}\oplus\mathbf{z}=(1,0)} D_{Y_1}(\mathbf{x})D_{Y_2}(\mathbf{y})D_{Y_3}(\mathbf{z}) = D(0,0)D(0,0)D(1,0)+$$
$$D(0,0)D(0,1)D(1,1) + \cdots + D(1,1)D(1,1)D(1,0) = \frac{132}{512} \approx 0.258.$$

$$D_{\Gamma^{(3)}}(1,1) = \sum_{\mathbf{x}\oplus\mathbf{y}\oplus\mathbf{z}=(1,1)} D_{Y_1}(\mathbf{x})D_{Y_2}(\mathbf{y})D_{Y_3}(\mathbf{z}) = D(0,0)D(0,0)D(1,1)+$$
$$D(0,0)D(0,1)D(1,0) + \cdots + D(1,1)D(1,1)D(1,1) = \frac{128}{512} = 0.25.$$

From the above calculations, we derive two features. First, to calculate one function value of $D_{\Gamma^{(l)}}$, we must sum $2^{(l-1)n}$ terms. That is, to calculate

$$D_{\Gamma^{(l)}}(\mathbf{x}) = \sum_{\mathbf{x}_1\oplus\mathbf{x}_2\oplus\cdots\oplus\mathbf{x}_l=\mathbf{x}} D_{Y_1}(\mathbf{x}_1)D_{Y_2}(\mathbf{x}_2)\cdots D_{Y_l}(\mathbf{x}_l), \tag{1}$$

the first $l-1$ terms $\mathbf{x}_1, \mathbf{x}_2, \cdots \mathbf{x}_{l-1}$ could be any value and the last term $\mathbf{x}_l$ is automatically determined by equation $\mathbf{x}_1 \oplus \mathbf{x}_2 \oplus \cdots \oplus \mathbf{x}_l = \mathbf{x}$. Because there is 2^n choices respectively, the total terms would be $2^{(l-1)n}$; however, it is difficult to calculate. Second, as l grows, $D_{\Gamma^{(l)}}$ tends to uniform distribution. The distance between the original distribution D and the uniform distribution I with respect to infinite norm $\|D - I\|_\infty$ of above example is $\frac{1}{8}$. However, we can observe that $\|D_{\Gamma^{(3)}} - I\|_\infty$ is $\frac{1}{128}$. As l grows, the terms that should be computed to calculate the function value grow rapidly; consequently, the impact of one

function value will decrease. Although this phenomenon seems natural, still, the following questions remain: Under what conditions does this convergence happen? How about the convergence rate? How can we prove the related results?

2.1. Entropy Accumulation with $n = 1$

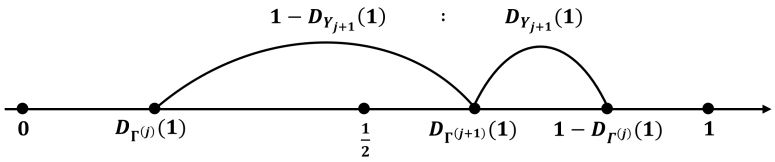
Let us consider the relatively simple case of $n = 1$ and Y_j following an independent and identical distribution (IID). In this case, all Y_j follow the same distribution $D(= D_{Y_j})$ and a recursive relationship $\Gamma^{(j+1)} = \Gamma^{(j)} \oplus Y_{j+1}$ is established. Thus, we can express $D_{\Gamma^{(j+1)}}(1)$ based on the following relationship:

$$D_{\Gamma^{(j+1)}}(1) = D_{\Gamma^{(j)}}(1)[1 - D_{Y_{j+1}}(1)] + [1 - D_{\Gamma^{(j)}}(1)]D_{Y_{j+1}}(1). \tag{2}$$

(2) is derived based on the property that the sum of two bits resulting in 1 can be obtained by adding 1 and 0, or 0 and 1. Moreover, $D_{\Gamma^{(j+1)}}(1)$ in (2) can be interpreted as a point where $D_{\Gamma^{(j)}}(1)$ and $1 - D_{\Gamma^{(j)}}(1)$ are internalized into $1 - D_{Y_{j+1}}(1) : D_{Y_{j+1}}(1)$. Because $D_{\Gamma^{(j)}}(1)$ and $1 - D_{\Gamma^{(j)}}(1)$ are symmetric about $x = \frac{1}{2}$, the condition $0 < D(1) < 1$ causes the convergence of $D_{\Gamma^{(j)}}(1)$ to $\frac{1}{2}$ (i.e., the maximum possible entropy) as j increases. Figure 5 illustrates the scenario.

$$\boldsymbol{D_{\Gamma^{(j+1)}}(1) = D_{\Gamma^{(j)}}(1)\left[1 - D_{Y_{j+1}}(1)\right] + \left[1 - D_{\Gamma^{(j)}}(1)\right]D_{Y_{j+1}}(1)}$$

$$\Downarrow$$



$$\Rightarrow \left|\frac{1}{2} - D_{\Gamma^{(j+1)}}(1)\right| \leq \left|\frac{1}{2} - D_{\Gamma^{(j)}}(1)\right|$$

Figure 5. Our entropy accumulation with $n = 1$.

A more specific formula exists to accurately illustrate this situation. The following lemma, often referred to as the "Piling Up Lemma," further details this [16].

Fact 1 (Piling Up Lemma [16])**.** *Let $Y_1, Y_2, \cdots, Y_l$ be independent one-bit random variables, and let $\Gamma^{(l)} := \sum_{j=1}^{l} Y_j$. Then,*

$$D_{\Gamma^{(l)}}(0) = \frac{1}{2} + 2^{l-1}\prod_{j=1}^{l}\left[D_{Y_j}(0) - \frac{1}{2}\right].$$

Because $0 < |D_{Y_j}(0) - \frac{1}{2}| < \frac{1}{2}$ for each j, $2^{l-1}\prod_{j=1}^{l}\left[D_{Y_j}(0) - \frac{1}{2}\right]$ converges to 0 and $D_{\Gamma^{(l)}}(0)$ converges to $1/2$ as l approaches infinity.

This equation cannot be applied when n is greater than 2. When $n = 2$ or more, the probability that each bit produces 0 or 1 converges to $1/2$; however, we cannot sum up the min-entropies of each position to calculate the total min-entropy because it is allowed only when all bits are independent of each other. Therefore, a new method is required for addressing these problems.

2.2. Convolution and Discrete Fourier Transform

In this subsection, we describe techniques applicable in the special case where n equals 1, as well as in more general cases. First, we reformulated the problem using the concept of convolution.

Definition 1 (Convolution)**.** *The Convolution of $f, g \in \mathcal{F}(\mathbb{Z}_m^n)$ is defined as*

$$f * g(\mathbf{x}) := \sum_{\mathbf{y} \in \mathbb{Z}_m^n} f(\mathbf{x} - \mathbf{y}) g(\mathbf{y}).$$

For the entropy accumulation problem of interest, $m = 2$. Using the language of convolution, (1) becomes $D_{\Gamma^{(l)}} = D_{Y_1} * D_{Y_2} * \cdots * D_{Y_l}$. The entropy accumulation problem is reduced to a problem of handling this convolution. Fortunately, there exists a mathematical concept, the "Fourier Transform", that harmonizes well with convolution.

Definition 2 (Discrete Fourier Transform)**.** *The Discrete Fourier Transform of $f \in \mathcal{F}(\mathbb{Z}_m^n)$ is defined as*

$$\widehat{f}(\mathbf{t}) := \sum_{\mathbf{x} \in \mathbb{Z}_m^n} f(\mathbf{x}) e^{-\frac{2\pi i}{m} \mathbf{x} \cdot \mathbf{t}}.$$

A Discrete Fourier Transform is the mapping from $\mathcal{F}(\mathbb{Z}_m^n)$ to $\mathcal{F}(\mathbb{Z}_m^n)$. In fact, this transform is one-to-one mapping. Proposition 1 supports this.

Lemma 1. *If $\mathbf{t} \neq \mathbf{0}$, $\sum_{\mathbf{x} \in \mathbb{Z}_m^n} e^{-\frac{2\pi i}{m} \mathbf{x} \cdot \mathbf{t}} = 0$.*

Proof. First, note that

$$\sum_{\mathbf{x} \in \mathbb{Z}_m^n} e^{-\frac{2\pi i}{m} \mathbf{x} \cdot \mathbf{t}} = \sum_{\mathbf{x} \in \mathbb{Z}_m^n} e^{-\frac{2\pi i}{m} \mathbf{x} \cdot \mathbf{t} (\mathrm{mod} m)}.$$

We define $\phi_{\mathbf{t}} : \mathbb{Z}_m^n \to \mathbb{Z}_m$ as:

$$\phi_{\mathbf{t}}(\mathbf{x}) := \mathbf{x} \cdot \mathbf{t} (\mathrm{mod} m).$$

Then, $\phi_{\mathbf{t}}$ is a homomorphism :

$$\phi_{\mathbf{t}}(\mathbf{x} + \mathbf{y}) = (\mathbf{x} + \mathbf{y}) \cdot \mathbf{t} (\mathrm{mod} m) = \mathbf{x} \cdot \mathbf{t} (\mathrm{mod} m) + \mathbf{y} \cdot \mathbf{t} (\mathrm{mod} m) = \phi_{\mathbf{t}}(\mathbf{x}) + \phi_{\mathbf{t}}(\mathbf{y}).$$

As $\mathbf{t} \neq \mathbf{0}$, $\phi_{\mathbf{t}}^{-1}(0) \neq \mathbb{Z}_m^n$. Therefore, for every $s \in \mathbb{Z}_m$, $\phi_{\mathbf{t}}^{-1}(s)$ contains the same number of elements. Let the number of elements be N, then

$$\sum_{\mathbf{x} \in \mathbb{Z}_m^n} e^{-\frac{2\pi i}{m} \mathbf{x} \cdot \mathbf{t}} = \sum_{\mathbf{x} \in \mathbb{Z}_m^n} e^{-\frac{2\pi i}{m} \phi_{\mathbf{t}}(\mathbf{x})} = N \sum_{s \in \mathbb{Z}_m} e^{-\frac{2\pi i}{m} s} = 0.$$

The last equality holds because each $e^{-\frac{2\pi i}{m} s}$ is the root of the complex equation $z^m - 1 = 0$. □

Lemma 2. *Let f be the element in $\mathcal{F}(\mathbb{Z}_m^n)$ and $\widehat{f}$ be the Fourier Transform function. Then,*

$$\frac{1}{m^n} \sum_{\mathbf{t} \in \mathbb{Z}_m^n} \widehat{f}(\mathbf{t}) e^{\frac{2\pi i}{m} \mathbf{x} \cdot \mathbf{t}} = f(\mathbf{x}).$$

Proof. By Lemma 1, $\sum_{\mathbf{t}\in\mathbb{Z}_m^n} f(\mathbf{s})e^{-\frac{2\pi i}{m}(\mathbf{x}-\mathbf{s})\cdot\mathbf{t}} = 0$ if $\mathbf{s} \neq \mathbf{x}$. Therefore,

$$\frac{1}{m^n}\sum_{\mathbf{s}\in\mathbb{Z}_m^n}\sum_{\mathbf{t}\in\mathbb{Z}_m^n} f(\mathbf{s})e^{-\frac{2\pi i}{m}(\mathbf{x}-\mathbf{s})\cdot\mathbf{t}} = \frac{1}{m^n}\sum_{\mathbf{t}\in\mathbb{Z}_m^n} f(\mathbf{x})e^{-\frac{2\pi i}{m}(\mathbf{x}-\mathbf{x})\cdot\mathbf{t}} = \frac{1}{m^n}\sum_{\mathbf{t}\in\mathbb{Z}_m^n} f(\mathbf{x}) = f(\mathbf{x}). \quad \square$$

Proposition 1. *Let f and g be the elements of $\mathcal{F}(\mathbb{Z}_m^n)$. If $\widehat{f} = \widehat{g}$, $f = g$.*

Proof. We assume that $\widehat{f} = \widehat{g}$. Then,

$$f(\mathbf{x}) = \frac{1}{m^n}\sum_{\mathbf{t}\in\mathbb{Z}_m^n} \widehat{f}(\mathbf{t})e^{\frac{2\pi i}{m}\mathbf{x}\cdot\mathbf{t}} = \frac{1}{m^n}\sum_{\mathbf{t}\in\mathbb{Z}_m^n} \widehat{g}(\mathbf{t})e^{\frac{2\pi i}{m}\mathbf{x}\cdot\mathbf{t}} = g(\mathbf{x})$$

holds for every $\mathbf{x} \in \mathbb{Z}_m^n$ from Lemma 2. $\square$

The following theorem asserts that the convolution product of functions is represented as a multiplication in the transformed space. This plays a significant role in proving our main theorem.

Proposition 2. *Let f and g be the elements of $\mathcal{F}(\mathbb{Z}_m^n)$. Then, $\widehat{f * g} = \widehat{f}\widehat{g}$.*

Proof. By the definitions of convolution and Discrete Fourier Transform,

$$\begin{aligned}
\widehat{f * g}(\mathbf{t}) &= \sum_{\mathbf{x}\in\mathbb{Z}_m^n}\sum_{\mathbf{y}\in\mathbb{Z}_m^n} f(\mathbf{x}-\mathbf{y})g(\mathbf{y})e^{-\frac{2\pi i}{m}\mathbf{x}\cdot\mathbf{t}} \\
&= \sum_{\mathbf{x}\in\mathbb{Z}_m^n}\sum_{\mathbf{y}\in\mathbb{Z}_m^n} f(\mathbf{x}-\mathbf{y})e^{-\frac{2\pi i}{m}(\mathbf{x}-\mathbf{y})\cdot\mathbf{t}}g(\mathbf{y})e^{-\frac{2\pi i}{m}\mathbf{y}\cdot\mathbf{t}} \\
&= \sum_{\mathbf{x}\in\mathbb{Z}_m^n} f(\mathbf{x}-\mathbf{y})e^{-\frac{2\pi i}{m}(\mathbf{x}-\mathbf{y})\cdot\mathbf{t}}\sum_{\mathbf{y}\in\mathbb{Z}_m^n} g(\mathbf{y})e^{-\frac{2\pi i}{m}\mathbf{y}\cdot\mathbf{t}} \\
&= \sum_{\mathbf{x}\in\mathbb{Z}_m^n} f(\mathbf{x})e^{-\frac{2\pi i}{m}\mathbf{x}\cdot\mathbf{t}}\sum_{\mathbf{y}\in\mathbb{Z}_m^n} g(\mathbf{y})e^{-\frac{2\pi i}{m}\mathbf{y}\cdot\mathbf{t}} \\
&= \widehat{f}(\mathbf{t})\widehat{g}(\mathbf{t}). \quad \square
\end{aligned}$$

To intuitively determine why $\Gamma^{(l)}$ converges to a uniform distribution, we must understand both the properties of the Discrete Fourier Transform applied to the distribution and the Discrete Fourier Transform of a uniform distribution.

Proposition 3. *Let D be an arbitrary probability distribution and I be a uniform distribution of $\mathbb{Z}_m^n$. Then,*

(i) *For all $\mathbf{t} \in \mathbb{Z}_m^n$, $|\widehat{D}(\mathbf{t})| \leq 1$.*
(ii) *$\widehat{D}(\mathbf{0}) = 1$.*
(iii) *For all $\mathbf{t} \in \mathbb{Z}_m^n$, $\widehat{I}(\mathbf{t}) = \delta_{\mathbf{t},\mathbf{0}}$.*

The symbols $\delta_{\mathbf{t},\mathbf{0}}$ denote the Kronecker delta. The Kronecker delta is defined as 1 when $\mathbf{t}$ is zero vector and 0 for all other $\mathbf{t}$.

Proof. Proof of (a) :

$$|\widehat{D}(\mathbf{t})| = |\sum_{\mathbf{x}\in\mathbb{Z}_m^n} D(\mathbf{x})e^{-\frac{2\pi i}{m}\mathbf{x}\cdot\mathbf{t}}| \leq \sum_{\mathbf{x}\in\mathbb{Z}_m^n} |D(\mathbf{x})e^{-\frac{2\pi i}{m}\mathbf{x}\cdot\mathbf{t}}| = \sum_{\mathbf{x}\in\mathbb{Z}_m^n} D(\mathbf{x}) = 1.$$

Proof of (b) :

$$\widehat{D}(\mathbf{0}) = \sum_{\mathbf{x}\in\mathbb{Z}_m^n} D(\mathbf{x})e^{-\frac{2\pi i}{m}\mathbf{x}\cdot\mathbf{0}} = \sum_{\mathbf{x}\in\mathbb{Z}_m^n} D(\mathbf{x}) = 1.$$

Proof of (c): We know from part (a) that $\widehat{I}(\mathbf{0}) = 1$. For the remaining $\mathbf{t} \neq \mathbf{0}$,

$$\widehat{I}(\mathbf{t}) = \sum_{\mathbf{x} \in \mathbb{Z}_m^n} I(\mathbf{x}) e^{-\frac{2\pi i}{m} \mathbf{x} \cdot \mathbf{t}} = \frac{1}{m^n} \sum_{\mathbf{x} \in \mathbb{Z}_m^n} e^{-\frac{2\pi i}{m} \mathbf{x} \cdot \mathbf{t}} = 0.$$

The last equality is based on Lemma 1. □

From Proposition 2, we have $\widehat{D_{\Gamma^{(l)}}} = \widehat{D_{Y_1}} \widehat{D_{Y_2}} \cdots \widehat{D_{Y_l}}$. By Proposition 3, $\widehat{D_{\Gamma^{(l)}}}(\mathbf{0}) = 1$, whereas for $\mathbf{t} \neq \mathbf{0}$, the value of $\widehat{D_{\Gamma^{(l)}}}(\mathbf{t})$ approaches 0 as l increases. Specifically, $\widehat{D_{\Gamma^{(l)}}}$ converges to $\delta_{\mathbf{t},\mathbf{0}}$ as l increases. Since the Discrete Fourier Transform is a one-to-one function by Proposition 1, we can infer that $D_{\Gamma^{(l)}}$ converges to I as l increases.

2.3. Main Theorem

In the previous subsection, we confirmed that $D_{\Gamma^{(l)}}$ converges to a uniform distribution I as l increases. In this subsection, we present a solution to the entropy accumulation problem based on this approach. Specifically, we aim to find a condition for the random variable Y_j and a value for l such that $D_{\Gamma^{(l)}}$ achieves a specific min-entropy. The following theorem is one of the main results of our study:

Theorem 1. *Let $Y_1, Y_2, \cdots Y_l$ be independent n-bit random variables, and $\Gamma^{(l)} := Y_1 \oplus Y_2 \oplus \cdots \oplus Y_l$. We define $\omega := \min\{\|D_{Y_1}\|_{min}, \|D_{Y_2}\|_{min}, \cdots, \|D_{Y_l}\|_{min}\}$. Then,*

$$H_{min}(D_{\Gamma^{(l)}}) \geq n - \log_2 \left[1 + (2^n - 1)(1 - 2^n\omega)^l\right] \approx n - \frac{1}{\ln 2}(2^n - 1)(1 - 2^n\omega)^l.$$

Note that the condition for n-bit random variables $Y_1, Y_2, \cdots Y_l$ is not an IID. Because the above theorem only requires the independence of random variables without the condition of identical distribution, it can be effectively applied when using parallel entropy sources. We provide the proof of Theorem 1.

Proof. For any function $f \in L(\mathbb{Z}_2^n)$, we have

$$\widehat{f}(\mathbf{t}) = \sum_{\mathbf{x} \in \mathbb{Z}_2^n} f(\mathbf{x}) e^{\pi i (\mathbf{x} \cdot \mathbf{t})} = \sum_{\mathbf{x} \in \mathbb{Z}_2^n} f(\mathbf{x}) (-1)^{\mathbf{x} \cdot \mathbf{t}}, \tag{3}$$

$$\frac{1}{2^n} \sum_{\mathbf{t} \in \mathbb{Z}_m^n} \widehat{f}(\mathbf{t}) e^{\pi i (\mathbf{x} \cdot \mathbf{t})} = \frac{1}{2^n} \sum_{\mathbf{t} \in \mathbb{Z}_m^n} \widehat{f}(\mathbf{t}) (-1)^{\mathbf{x} \cdot \mathbf{t}} = f(\mathbf{x}). \tag{4}$$

This is obtained from Lemma 2 with $m = 2$. Using the function $\phi_{\mathbf{t}}$ of Lemma 2, (3) and (4) can be written as

$$\widehat{f}(\mathbf{t}) = \sum_{\mathbf{x} \in \mathbb{Z}_2^n} f(\mathbf{x}) e^{\pi i (\mathbf{x} \cdot \mathbf{t})} = \sum_{\mathbf{x} \in \mathbb{Z}_2^n} f(\mathbf{x}) (-1)^{\mathbf{x} \cdot \mathbf{t}} = \sum_{\mathbf{x} \in \phi_{\mathbf{t}}^{-1}(0)} f(\mathbf{x}) - \sum_{\mathbf{x} \in \phi_{\mathbf{t}}^{-1}(1)} f(\mathbf{x}), \tag{5}$$

$$\frac{1}{2^n} \sum_{\mathbf{t} \in \mathbb{Z}_2^n} \widehat{f}(\mathbf{t}) e^{\pi i (\mathbf{x} \cdot \mathbf{t})} = \frac{1}{2^n} \sum_{\mathbf{t} \in \mathbb{Z}_2^n} \widehat{f}(\mathbf{t}) (-1)^{\mathbf{x} \cdot \mathbf{t}} = \frac{1}{2^n} \left[\sum_{\mathbf{t} \in \phi_{\mathbf{x}}^{-1}(0)} \widehat{f}(\mathbf{t}) - \sum_{\mathbf{t} \in \phi_{\mathbf{x}}^{-1}(1)} \widehat{f}(\mathbf{t}) \right] = f(\mathbf{x}). \tag{6}$$

We apply (5) to each D_{Y_j} with $\mathbf{t} \neq \mathbf{0}$. Because $\sum_{\mathbf{x}\in\mathbb{Z}_2^n} D_{Y_j}(\mathbf{x}) = 1$ and $D_{Y_j}(\mathbf{x}) > \omega > 0$,

$$\begin{aligned}\left|\widehat{D_{Y_j}}(\mathbf{t})\right| &= \left|\sum_{\mathbf{x}\in\phi_{\mathbf{t}}^{-1}(0)} D_{Y_j}(\mathbf{x}) - \sum_{\mathbf{x}\in\phi_{\mathbf{t}}^{-1}(1)} D_{Y_j}(\mathbf{x})\right| \\ &= \left|\sum_{\mathbf{x}\in\phi_{\mathbf{t}}^{-1}(0)} \left[D_{Y_j}(\mathbf{x}) - \omega\right] - \sum_{\mathbf{x}\in\phi_{\mathbf{t}}^{-1}(1)} \left[D_{Y_j}(\mathbf{x}) - \omega\right]\right| \\ &\leq \left|\sum_{\mathbf{x}\in\mathbb{Z}_2^n} \left[D_{Y_j}(\mathbf{x}) - \omega\right]\right| = 1 - 2^n\omega.\end{aligned} \tag{7}$$

From Theorems (2) and (7),

$$\begin{aligned}\left|D_{\Gamma^{(l)}}(\mathbf{t})\right| &= \frac{1}{2^n}\left|\sum_{\mathbf{t}\in\phi_{\mathbf{x}}^{-1}(0)} \widehat{D_{\Gamma^{(l)}}}(\mathbf{t}) - \sum_{\mathbf{t}\in\phi_{\mathbf{x}}^{-1}(1)} \widehat{D_{\Gamma^{(l)}}}(\mathbf{t})\right| \\ &\leq \frac{1}{2^n}\sum_{\mathbf{t}\in\mathbb{Z}_2^n}\left|\widehat{D_{\Gamma^{(l)}}}(\mathbf{t})\right| = \frac{1}{2^n}\sum_{\mathbf{t}\in\mathbb{Z}_2^n}\prod_{j=1}^{l}\left|\widehat{D_{Y_j}}(\mathbf{t})\right| \\ &= \frac{1}{2^n}\left[\prod_{j=1}^{l}\left|\widehat{D_{Y_j}}(\mathbf{0})\right| + \sum_{\mathbf{t}\in\mathbb{Z}_2^n/\{\mathbf{0}\}}\prod_{j=1}^{l}\left|\widehat{D_{Y_j}}(\mathbf{t})\right|\right] \\ &\leq \frac{1}{2^n}\left[1 + (2^n - 1)(1 - 2^n\omega)^l\right].\end{aligned}$$

Therefore,

$$\begin{aligned}H_{min}(D_{\Gamma^{(l)}}) &= \max\{-\log_2\left|D_{\Gamma^{(l)}}(\mathbf{t})\right| : \mathbf{t} \in \mathbb{Z}_2^n\} \\ &\geq n - \log_2\left[1 + (2^n - 1)(1 - 2^n\omega)^l\right] \\ &\approx n - \frac{1}{\ln 2}(2^n - 1)(1 - 2^n\omega)^l.\end{aligned}$$

The final approximation is based on the Taylor theorem. □

3. Applying Theorem 1 to Image Sensor-Based RNG

In this section, we describe the process of applying Theorem 1 to an image sensor-based random number generator. First, we describe the process of generating the input sequences from the entropy sources of the image sensor. Subsequently, we verify whether the generated input sequences satisfy the assumptions of Theorem 1. Next, we establish the lower bound for the min-entropy, which is considered secure based on three standards. Then, we provide the theoretical number of iterations required to achieve a min-entropy higher than the established lower bound. Furthermore, we validate our theory using experimental results. Finally, we estimate the entropy accumulation speed based on frames per second (FPS) of the image sensors used. Thereafter, we compare and analyze the random number generation speed of our system with that of the ID Quantique's QRNG chip.

3.1. Image Sensor-Based RNG

We use the 'PV 4209K' image sensor, which utilizes 11,520 optical black pixels (OBP) as physical entropy sources. Each OBP of the image sensor transmits 2-bit data to a PC. The PC sequentially stores the 2-bit data transmitted by the multiple OBPs. See Figure 6.

0	1	0	1	2	0	3	2
1	0	2	2	0	2	2	0
2	2	3	1	2	1	3	0
2	0	1	0	2	1	0	2
1	0	0	1	1	0	0	0
0	0	0	0	2	2	2	0
2	1	2	0	2	0	0	1
1	0	1	2	2	3	0	0

Figure 6. Data transmission process of image sensor.

3.2. Experimentation Process for Entropy Accumulation

Before describing the entropy accumulation experiments, we use the following notation:

- W_i : A random variable corresponding to the 2-bit data of the i-th optical black pixel (OBP). "If the value of i reaches the last pixel (11,520), the next value of i refers to the first pixel."
- $Y_j := W_{4j-3} \| W_{4j-2} \| W_{4j-1} \| W_{4j}$. For example, if $W_1 = (0,1)$, $W_2 = (1,1)$, $W_3 = (0,0)$, $W_4 = (1,0)$, Y_1 becomes $Y_1 = (0,1,1,1,0,0,1,0)$.
- $\Gamma_k^{(l)} := \sum_{j=(kl-l+1)}^{kl} Y_j$. This refers to the k-th output sequence, which is generated by adding (XOR) l input sequences.

To experimentally validate entropy accumulation, we utilized the verification method outlined in [17]. Ref. [17] is a min-entropy estimation tool, which estimates the min-entropy of output sequences by collecting 1,000,000 n-bit output sequences. However, there is a requirement that the value of n must be at least 8. Therefore, to satisfy this condition, we created new 8-bit data Y_j by concatenating four 2-bit datasets W_{4j-3}, W_{4j-2}, W_{4j-1}, W_{4j}, and Y_j becomes an input sequence. This process is shown in Figure 7.

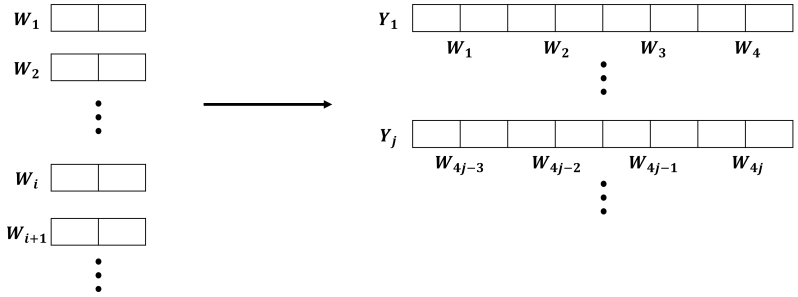


Figure 7. Concatenation of 2-bit data to form 8-bit input sequence.

After setting Y_j, we select the XOR operation iteration number $(l-1)$ and accumulate $\Gamma_k^{(l)}$ in the entropy pool. This process is illustrated in Figure 8.

The number l is determined using Theorem 1. After collecting $\Gamma_k^{(l)}$ $(1 \leq k \leq 1{,}000{,}000)$ in the entropy pool, we use [17] to verify the min-entropy.

3.3. Setting Lower Bound of Min-Entropy

We provide three evaluation criteria that can be used to determine the lower bound of the min-entropy, which a true random number generator must exceed, acquired through the entropy accumulation process.

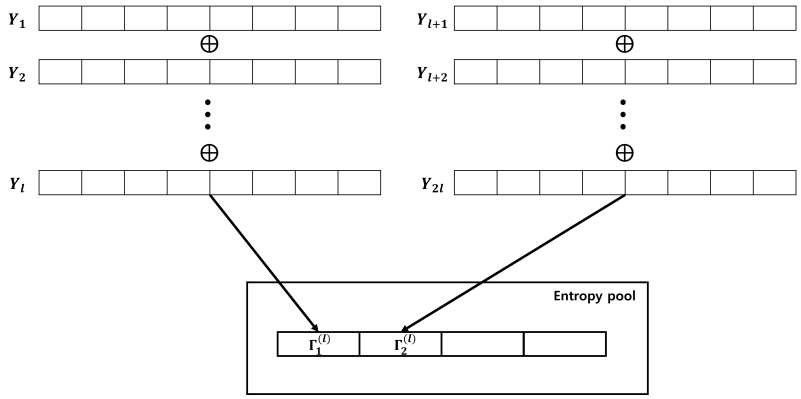


Figure 8. Accumulating entropy source using XOR operation.

3.3.1. Maximum Value of the Most Common Value Estimate

In [17], when the output sequences are determined to follow the IID, the Most Common Value Estimate is assumed to be the min-entropy of the output sequences [17]. However, there is an upper bound on the min-entropy value in the Most Common Value Estimate. Regardless of the output sequences used for the test, this upper bound cannot be exceeded. The Most Common Value Estimate estimates the min-entropy as described in Algorithm 1.

Algorithm 1 Most Common Value Estimate.

Input: $S = (s_1, \ldots, s_L)$, L : length of S, $S_i \in \{0,1\}^n$ $(1 \leq i \leq L)$
Output: min-entropy of dataset S : H_{min}

1: Calculate the mode of S. We denote this value by $MODE$
2: $\hat{p} = \frac{MODE}{L}$
3: $p_u = \min\left(1, \hat{p} + 2.576\sqrt{\frac{\hat{p}(1-\hat{p})}{L-1}}\right)$
4: $H_{min} = -\log_2 p_u$

To compute the upper bound of the Most Common Value Estimate for an 8-bit dataset S, where the length of S is 1,000,000, the mode of S should be one. That is, $\hat{p} = 1/256$. Using $\hat{p}$, we obtain $H_{min} = 7.94$. Therefore, it is reasonable for the lower bound of the min-entropy to not exceed 7.94.

3.3.2. True Random 8-Bit in [17]

Ref. [17] provides True Random Data in 1-bit, 4-bit, and 8-bit units as samples for evaluating min-entropy. From Figure 9, it can be confirmed that the min-entropy of true random 8-bit data in [17] is approximately 7.86.

```
root@940bfbc87d09:/90b/cpp# ./ea_iid -i -a truerand_8bit.bin 8
Calculating baseline statistics...
H_original: 7.865118
H_bitstring: 0.998199
min(H_original, 8 X H_bitstring): 7.865118

** Passed chi square tests

** Passed length of longest repeated substring test

Beginning initial tests...
Beginning permutation tests... these may take some time
** Passed IID permutation tests
```

Figure 9. Min-entropy of true random 8-bit.

3.3.3. Criterion of Min-Entropy by BSI AIS 20/31 [18]

The Federal Office for Information Security in Germany (Bundesamt für Sicherheit in der Informationstechni, BSI) asserts in the AIS 20/31 document that the output sequences of a cryptographic random number generator should have a min-entropy of 0.98 per bit. In the case of 8-bit data, 7.84 becomes the lower bound of entropy.

From the three criteria mentioned above, we have chosen 7.86 as the min-entropy lower bound. This value, which is smaller than 7.94 and more stringent than 7.84, appears to be a valid choice for the lower bound.

3.4. Applying Theorem 1 to Input Sequences

In this subsection, we use Theorem 1 and obtain l. Instead of directly applying Theorem 1 to Y_j, we employ a "divide and conquer" approach to compute the total min-entropy. Before explaining this strategy, note that each W_i can be regarded as an independent random variable. This assumption is reasonable because all pixels can be considered independent entropy sources.

Because we conducted the experiment with eight bits, n must be eight when applying Theorem 1. However, this results in the following problem: the required number l is exceedingly large. To address this issue, we exploit the fact that Y_j is constructed by concatenating four independent entropy sources. $\Gamma_k^{(l)}$ is generated by considering the XOR of l instances in Y_j. Therefore, if we break down $\Gamma_k^{(l)}$ into two bits, then $\Gamma_k^{(l)}$ can be considered as four concatenations of the XOR of l instances of W_i. This can be expressed by the following formula:

$$\Gamma_k^{(l)} = \sum_{j=(kl-l+1)}^{kl} W_{4j-3} \| W_{4j-2} \| W_{4j-1} \| W_{4j} =$$
$$\left(\sum_{j=(kl-l+1)}^{kl} W_{4j-3} \right) \| \left(\sum_{j=(kl-l+1)}^{kl} W_{4j-2} \right) \| \left(\sum_{j=(kl-l+1)}^{kl} W_{4j-1} \right) \| \left(\sum_{j=(kl-l+1)}^{kl} W_{4j} \right).$$

The four parts of $\Gamma_k^{(l)}$ are independent of each other. Therefore, we calculated the total min-entropy by individually determining the min-entropy for each of the four parts and then summing them up. If the min-entropy of each 2-bit segment of $\Gamma_k^{(l)}$ is greater than 1.965, $H_{min}(\Gamma_k^{(l)})$ will be greater than 7.86. Theorem 1 is applied to the four 2-bit segments of $\Gamma_k^{(l)}$. To apply Theorem 1, the following two conditions must be satisfied. The first pertains to the independence of W_i. The second condition states that the value of ω should exceed zero. For the first condition, we established that each W_i could be regarded as an independent random variable. For the second condition, we can estimate the value of ω by analyzing the probability distribution of the data transmitted by each OBP. We constructed the probability distribution of each W_i using 2-bit data transmitted by each of the 11,520 OBPs over 2000 transmissions. From the obtained distribution, we can confirm that the value of $\|D_{W_i}\|_{min}$ is greater than 0.075 for all i. Figure 10 illustrates the probability distribution of four randomly selected OBPs. It can be observed that each number has appeared at least 150 times.

Therefore, we estimate that ω is at least 0.075. From Theorem 1, the inequality

$$2 - \frac{1}{\ln 2}(2^2 - 1)(1 - 2^2\omega)^l \geq 1.965$$

provides the number l necessary for the min-entropy of each 2-bit segment of $\Gamma_k^{(l)}$ to exceed 1.965.

$$
\begin{aligned}
&2 - \frac{1}{\ln 2}(2^2 - 1)(1 - 2^2\omega)^l \geq 1.965 \\
&\Rightarrow \frac{0.035 \cdot \ln 2}{3} \geq (0.7)^l \\
&\Rightarrow \frac{\ln\left(\frac{0.035 \cdot \ln 2}{3}\right)}{\ln(0.7)} \leq l \\
&\Rightarrow 15.845 \leq l.
\end{aligned}
$$

From the last inequality, we can conclude that if we use 15 XOR operations to create $\Gamma_k^{(l)}$, $H_{min}(\Gamma_k^{(l)})$ exceeds 7.86.

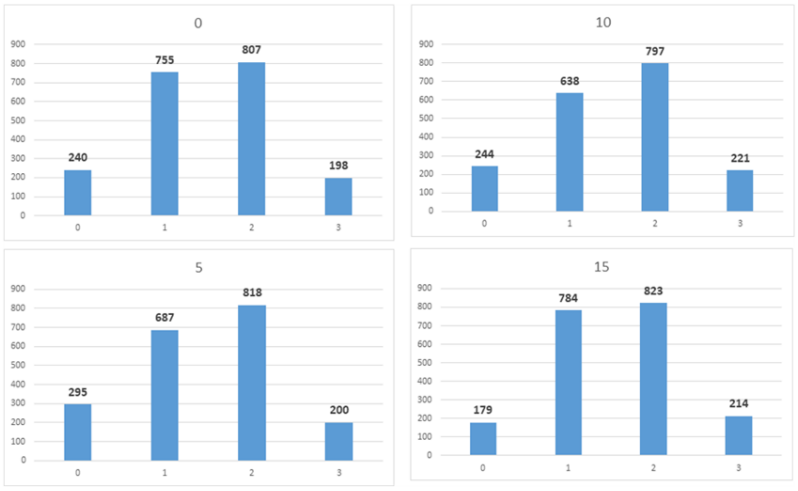


Figure 10. Probability distribution of each OBP.

3.5. Experimental Result

We describe the experimental validation of the results in Section 3.4. We predicted that through 15 XOR operations for entropy accumulation, more than 7.86-bit of entropy per 8-bit would be guaranteed. Upon conducting actual experiments, it was confirmed that even with only four XOR operations, more than 7.86 min-entropy per 8-bit was accomplished. Table 1 presents the experimental results.

Table 1. Min-entropy values corresponding to number of XOR operation repetitions.

l	i	j	k	Min-Entropy per 8-Bit
1	4,000,000	1,000,000	1,000,000	3.305
2	8,000,000	2,000,000	1,000,000	7.115
3	12,000,000	3,000,000	1,000,000	7.700
4	16,000,000	4,000,000	1,000,000	7.852
5	20,000,000	5,000,000	1,000,000	7.864

The experimental results are analyzed as follows. When $l = 1$, it refers to the case where the XOR operation is not used, and the min-entropy per 8-bit is 3.305, which is lower than the min-entropy calculated based on the probability distribution of the 2-bit from the pixels.

Min-entropy depends on the maximum value of the probability distribution function. By observing the 2-bit probability distribution, we confirmed that the maximum value of

the probability distribution function was 0.425. This can also be confirmed from Figure 10, where all the numbers (0, 1, 2, 3) in the four randomly selected distributions do not exceed 850. Therefore, when calculating the min-entropy of 2-bit, the lower bound of $H_{min}(D_{W_i})$ is 1.2344, and the lower bound of $H_{min}(D_{Y_j})$ created by concatenating the four D_{W_i} is 4.9378.

The min-entropy estimated through the probability distribution is lower than that estimated by [17] because of the conservative measurement method of [17]. In [17], it is determined whether the output sequence follows the IID track or the non-IID track before measuring min-entropy. If the output sequence follows the IID track, the min-entropy measured by the most common value estimation method becomes the final min-entropy of the output sequence. However, if the output sequence follows the non-IID track, the smallest value among the min-entropies measured by the other 10 methods, including the Most Common Value Estimate, becomes the final min-entropy of the output sequence. For $l = 1$, it was confirmed that the original output sequence followed a non-IID track. Therefore, the min-entropy value estimated by [17] was smaller than that calculated based on the probability distribution.

As the value of l increases, it can be observed that the increment of min-entropy decreases. This is because the maximum possible min-entropy value is 7.94, and as it approaches this number, the amount of min-entropy that can be increased by increasing the number of XOR operations becomes small. The greatest change in the min-entropy value occurs when l changes from $l = 1$ to $l = 2$. This is because the output sequence follows the IID track if $l \geq 2$.

4. Comparing with Other Entropy Accumulations

In this section, we compare and analyze the entropy accumulation we developed with other entropy accumulations. The first comparison is with the Slow-Refresh used in IDQ QRNG. The second comparison target is the Fast-Refresh of Windows RNG. In comparing with the Slow-Refresh, we focus on the differences in the accumulation mechanism. In comparing with the Fast-Refresh, the analysis is carried out by calculating the iteration number of operations of Fast-Refresh in our experimental environment.

4.1. Comparing with the Slow-Refresh of IDQ QRNG

IDQ QRNG uses matrix multiplication as the entropy accumulation function [19,20]. For the analysis of this method, we first explain the theoretical background of entropy accumulation and the "Leftover Hash Lemma" and then explain the entropy accumulation of IDQ QRNG. We also check whether the entropy accumulation of IDQ QRNG meets the conditions of the Leftover Hash Lemma. Lastly, we summarize the differences between our entropy accumulation and that of IDQ QRNG.

4.1.1. Leftover Hash Lemma [3,19]

The Leftover Hash Lemma is a theorem that ensures the conversion of a low min-entropy input sequence into a high min-entropy output sequence. In order to state the Leftover Hash Lemma, we need two key concepts: the 2-universal hash family and statistical distance.

Definition 3 (Statistical distance)**.** *Let X and X' be random variables that take values in same set. The statistical distance between two probability distributions D_X and $D_{X'}$ is defined as*

$$\Delta_{D_X, D_{X'}} = \frac{1}{2}\|D_X - D_{X'}\|_1.$$

Definition 4 (2-universal hash family)**.** *Let Y be a random variable uniformly distributed over S. A family $\{f_s : T \to V\}_{s \in S}$ of hash functions is called 2-universal if for any distinct inputs $x \neq x'$*

$$Pr\left[f_Y(x) = f_Y(x')\right] \leq \frac{1}{|V|}.$$

Based on these definitions, we can state the Leftover Hash Lemma.

Fact 2 (Leftover Hash Lemma). *Let $\{f_s : T \to V\}_{s\in S}$ be the 2-universal hash family. Let X and Y be independent random variables, where Y is uniformly distributed over S, and X takes values in T. Let $U_{S\times V}$ be the uniform distribution on $S \times V$. Then,*

$$\Delta_{D_{(Y,f_Y(X))},U_{S\times V}} \leq 2^{-\frac{1}{2}(H_{min}(X)-\log_2|V|)}.$$

Corollary 1. *Under the same assumptions, we obtain that*

$$\begin{aligned} H_{min}(D_{f_Y(X)}) &\geq -\log_2\left(\frac{1}{|V|} + 2^{-\frac{1}{2}(H_{min}(X)-\log_2|V|)+1}\right) \\ &\approx \log_2|V| - \frac{|V|}{\ln 2}2^{-\frac{1}{2}(H_{min}(X)-\log_2|V|)+1} \\ &= \log_2|V| - \epsilon. \end{aligned}$$

Proof. Let U_V be the uniform distribution on V. By triangle inequality and Leftover Hash Lemma,

$$\begin{aligned} H_{min}(D_{f_Y(X)}) &= -\log_2\|D_{f_Y(X)}\|_\infty \\ &\geq -\log_2\left[\|U_V\|_\infty + \|D_{f_Y(X)} - U_V\|_\infty\right] \\ &\geq -\log_2\left[\|U_V\|_\infty + \|D_{f_Y(X)} - U_V\|_1\right] \\ &\geq -\log_2\left[\|U_V\|_\infty + \|D_{(Y,f_Y(X))} - U_{S\times V}\|_1\right] \\ &= -\log_2\left[\|U_V\|_\infty + 2\Delta_{D_{(Y,f_Y(X))},U_{S\times V}}\right] \\ &\geq -\log_2\left(\frac{1}{|V|} + 2^{-\frac{1}{2}(H_{min}(X)-\log_2|V|)+1}\right) \\ &\approx \log_2|V| - \frac{|V|}{\ln 2}2^{-\frac{1}{2}(H_{min}(X)-\log_2|V|)+1}. \end{aligned}$$

The last approximation is due to Taylor expansion at $x = \frac{1}{|V|}$. □

The Leftover Hash Lemma is effective in generating high-quality output sequences in the following situations:

(i) When $H_{min}(X)$ is smaller than $|T|$.
(ii) When $H_{min}(X)$ is significantly larger than $|V|$.
(iii) When $|S|$ is substantially smaller than $|V|$.

The key point here is the last condition. To generate output sequences using the Leftover Hash Lemma, a hash function must be uniformly selected from the 2-universal hash family. This means that random numbers are required to generate random numbers. If the size of the 2-universal hash family is small, a large amount of random numbers can be generated using a small amount of random numbers. Therefore, constructing a 2-universal hash family of small size is the key point in the use of the Leftover Hash Lemma.

Example 1. Suppose that $T = \{0,1\}^{900}$, $V = \{0,1\}^{100}$, $|S| = 2^{30}$, $H_{min}(X) = 550$, $H_{min}(Y) = 30$, then by Corollary 1, $H_{min}(D_{f_Y(X)}) \geq 100 - \frac{1}{\ln 2}2^{-124}$. This implies that if we have a low quality entropy source with a min-entropy of 550 out of 900, and a 2-universal hash family of size 2^{30}, we can leverage a 30-bit random number generator to produce nearly random 100-bit numbers.

4.1.2. Slow-Refresh of IDQ QRNG [19,20]

IDQ QRNG uses an $m \times n$ random matrix as the entropy accumulation function. This function transforms an input sequence of length m into an output sequence of length n. It

can also be readily proven that this collection of matrices forms a 2-universal hash family [4]. Using the notation from the Leftover Hash Lemma, this can be represented as

$$T = \{0,1\}^n,\ V = \{0,1\}^m,\ S = \{0,1\}^{m \times n}.$$

The min-entropy per bit of the quantum entropy source (which corresponds to $H_{min}(X)$ in our notation) is not disclosed. However, according to [19], the ϵ value is designed to be set to 2^{-100}. The random matrix is generated only once, and the mn elements that make up this matrix are generated using a 1-bit random number generator mn times. The 1-bit random number generator creates a single bit by collecting multiple bits from the digitized entropy source and performing an XOR operation. The more bits that are XORed, the higher the min-entropy of the generated 1-bit. This can be confirmed through the Piling Up Lemma. Figure 11 illustrates the entropy accumulation of IDQ QRNG.

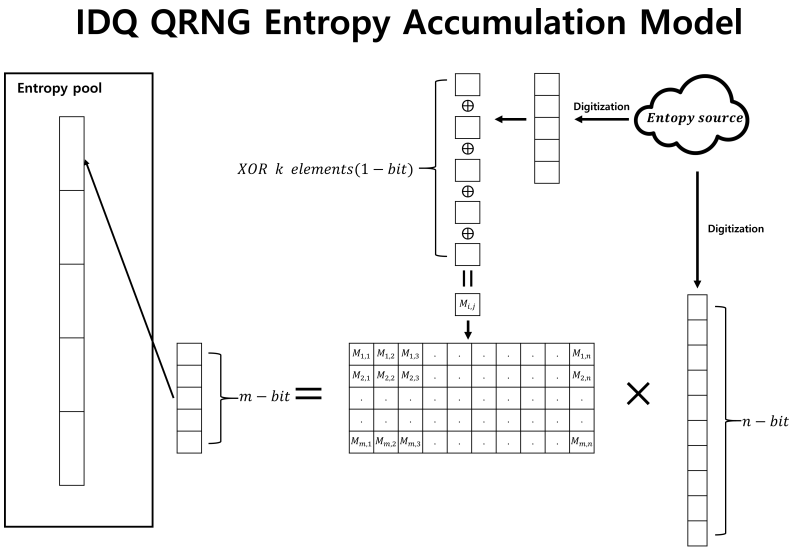


Figure 11. IDQ QRNG entropy accumulation model.

4.1.3. Limitations of IDQ QRNG Entropy Accumulation Model

At first glance, the entropy accumulation model of IDQ QRNG seems to generate random numbers based on the Leftover Hash Lemma, but this model has two inherent limitations.

The first point concerns the safety of the 1-bit random number generator used to create the random matrix. This generator should output 0 and 1 with a probability of 1/2. However, the min-entropy of the entropy source used in IDQ QRNG and the iteration number of XOR operations are not specified, which makes it impossible to confirm whether the random matrix was uniformly generated.

The second point is that the random matrix is generated just one time in IDQ QRNG, but this implementation seems an incomplete application of the Leftover Hash Lemma. If the random matrix is continuously used, the condition of independence between the consecutive m-bit output sequences is compromised. Thus, in this case, it is impossible to obtain the overall entropy of long output sequence by the Leftover Hash Lemma. For example, if we generate m-bit output sequences X_i's ($1 \leq i \leq 5$) by using the entropy accumulation of IDQ QRNG, then although for each $1 \leq i \leq 5$, $H_{min}(X_i) = m - \epsilon$, it does not always satisfy that $H_{min}(X_1 \| X_2 \| X_3 \| X_4 \| X_5) = 5(m - \epsilon)$. Therefore, the matrix should be updated whenever a new output sequence is generated in order to properly apply the Leftover Hash Lemma.

4.1.4. Differences between IDQ QRNG and Our Entropy Accumulation

The simple characteristic distinguishing Fast-Refresh from Slow-Refresh is the length of the input sequence. In our entropy accumulation, we generate the output sequence by XORing five 8-bit input sequences, that is, the total length of the input sequence is 40-bit with the 8-bit length output sequence. However, IDQ QRNG uses the input sequence of 1024-bit or 2048-bit, and the output sequences of 768-bit or 1792-bit, respectively [19]. The reason for such a long length of the input sequence is to adjust the ϵ of the Leftover Hash Lemma to about 2^{-100} [19].

The bit loss rate is another characteristic distinguishing Fast-Refresh from Slow-Refresh. In our entropy accumulation process, 32-bit are discarded from 40-bit of input sequence, resulting in a bit loss rate of 80%. On the other hand, in the IDQ QRNG entropy accumulation process, 256-bit are discarded from 1024-bit or 2048-bit of input sequence, resulting in bit loss rates of 25% or 12.5%, respectively. Through this, we can see that the Slow-Refresh handles a large number of bits at once but has a relatively low loss rate. However, a low bit loss rate results in a decrease in operation speed. The major differences are shown in Table 2.

Table 2. Major differences between our entropy accumulation and that of IDQ QRNG.

	Our Entropy Accumulation	**IDQ QRNG**
Refresh Type	Fast-Refresh	Slow-Refresh
Theoretical Background	Fourier Transform	Leftover Hash Lemma
Implementation Aspect	Simple XOR operations	Difficulty of implementing universal hash family
Input Sequence Length	40	1024 or 2048
Output Sequence Length	8	768 or 1792
Bit Loss Rate	80%	25% or 12.5%

4.2. Comparing with the Fast-Refresh of Windows RNG

We have already described how Fast-Refresh of Windows RNG works in the introduction. Hence, we describe the entropy accumulation theory in [2] and calculate the number of operations that must be iterated when applying it to the input sequence Y_j. First, we describe the 2-monotone distribution and the covering number, which are essential concepts in [2]. Thereafter, we present Theorem 5.2 of [2] (which is the main result of [2]) with our notation.

Next, we explain why [2] cannot be directly applied to our entropy accumulation model and suggest an additional S-box operation as a solution. With this additional operation, we can apply Theorem 5.2 of [2] and overcome the limitations of the original theory. Finally, we provide the theoretical number of operations necessary to guarantee a min-entropy of 7.86 per 8-bit when Theorem 5.2 of [2] is applied to the RNG.

4.2.1. Main Results of [2]

The covering number is used to measure the efficiency of the permutations used in entropy accumulation. The efficiency of entropy accumulation increased as the covering number of permutations decreased.

Definition 5 (Covering number)**.** *For a permutation* $\pi : \{0, 1, ..., n-1\} \to \{0, 1, ..., n-1\}$*, and an integer* $1 \leq k \leq n$*, the covering number* $C_{\pi,k}$ *is the smallest natural number* m *such that*

$$\{\pi^l(j) : 0 \leq j < k, 0 \leq l < m\} = \{0, 1, ..., n-1\}. \quad (8)$$

If no such m *exists, then* $C_{\pi,k} = \infty$*;*

Figure 12 shows the calculation of the number of coverings.

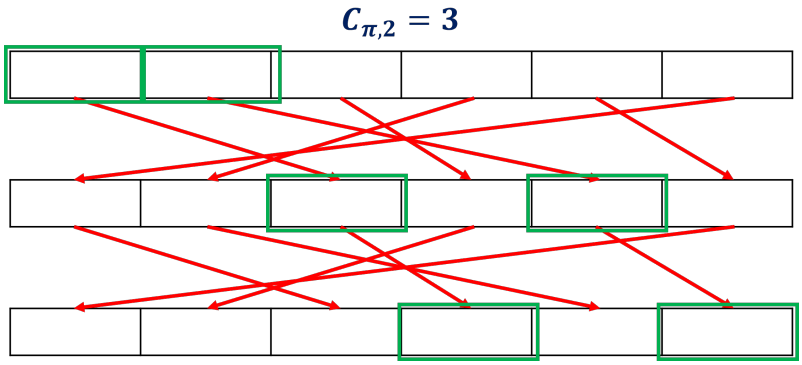


Figure 12. Illustration of covering number when $C_{\pi,2} = 3$. The covering number is 3 because all bits are covered using the permutation operation twice.

One special property of an entropy source in [2] is 2-monotone distribution. The definition is as follows:

Definition 6 (2-monotone distribution)**.** *The probability distribution D_X of an n-bit random variable X follows a 2-monotone distribution if its domain can be divided into two disjoint intervals, where D_X is a monotone function.*

A 2-monotone distribution has at least one inflection point (peak), because it is divided into two monotonic intervals.

Based on these definitions, we can state Theorem 5.2 of [2].

Theorem 2 (Theorem 5.2 of [2])**.** *Suppose that for independent n-bit random variables $Y_1, Y_2, ..., Y_l$, the probability distributions $D_{Y_1}, D_{Y_2}, ..., D_{Y_l}$ have a min-entropy of at least $k(\geq 2)$ and follow a 2-monotone distribution. Let $\pi : \{0, 1, ..., n-1\} \rightarrow \{0, 1, ..., n-1\}$ be a permutation and $m = C_{\pi,k'}$ be a covering number where $k' = \left\lfloor \frac{k}{2} \right\rfloor$. Let $\Lambda_\pi^{(l)} = A_\pi^{l-1}(Y_1) \oplus A_\pi^{l-2}(Y_2) \oplus \cdots \oplus Y_l$. Then, for any $l \geq m$,*

$$H_{min}(D_{\Lambda_\pi^{(l)}}) \geq n - \left(\left\lfloor \frac{n}{k'} \right\rfloor + 1 \right) \cdot log_2 \left(1 + 2^{k' - \left(\frac{k}{2}\right) \left\lfloor \frac{l}{m} \right\rfloor} \right) \approx n \left(1 - 2^{\frac{k}{2} - \frac{kl}{2m}} \right).$$

One can easily observe that as n increased, $H_{min}(D_{\Lambda_\pi^{(l)}})$ converged to n.

4.2.2. Windows RNG Entropy Accumulation without 2-Monotone Condition

Theorem 2 provides a min-entropy lower bound for $\Lambda_\pi^{(l)}$ with only three restrictions. The conditions under which the input sequences are independent, and the covering number of permutations is finite, are relatively easy to satisfy. However, it is challenging to satisfy the condition that all input sequences follow a 2-monotone distribution. In particular, for the image sensor entropy sources used, input sequences that follow a 2-monotone distribution are unattainable.

We generated Y_j by concatenating four 2-bit entropy sources: $W_{4j-3}, W_{4j-2}, W_{4j-1}, W_{4j}$. We used this method only for experimental verification (SP-800-90b requires at least 8-bit data), and $H_{min}(\Gamma^{(l)})$ was theoretically calculated by adding the four min-entropy values of the 2-bit segment of $\Gamma^{(l)}$(see Section 3.4). However, if we apply Theorem 2 to our input sequences, we cannot use the divide-and-conquer approach. This is because while D_{W_i} definitely satisfies a 2-monotone (as can be observed in Figure 10), Theorem 2 requires input sequences with a min-entropy of two or more. Note that the 2-bit entropy sources W_i cannot achieve this condition. For this reason, when applying Theorem 2 to our input sequences, we must use the concatenation method for theoretical, rather than experimental, reasons. However, if the input sequences are processed in a concatenated manner, it is impossible

to satisfy the 2-monotone assumption. We explain this using a 4-bit example. Figure 13 represents the probability distribution of 2-bit entropy sources W_1 and W_2 that follow a 2-monotone distribution. The probability distribution of $W_1 \| W_2$, created by concatenating the two entropy sources, is shown in Figure 14.

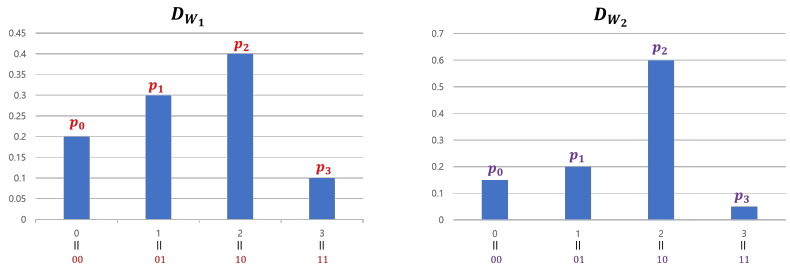


Figure 13. Example of two distributions that follow a 2-monotone distribution.

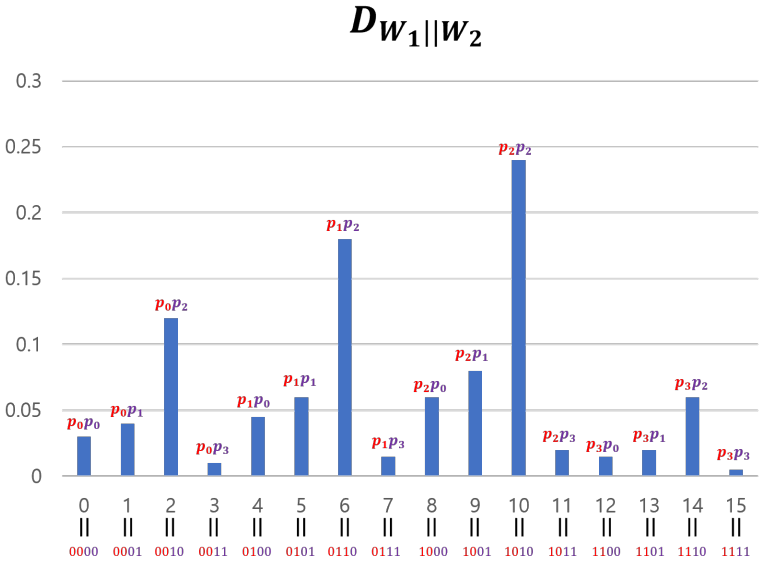


Figure 14. Distribution of concatenated entropy sources.

If we consider $\{4i-3, 4i-2, 4i-1, 4i\}$ as one group, there are four groups and the overall shape of $D_{W_1 \| W_2}$ is similar to that of D_{W_1}. However, the shape of each group's graph is similar to D_{W_2}. The shape of such a concatenated distribution tends to become more complex as the entropy sources become more concatenated. Therefore, inevitably, the distribution of the input sequences assumes a shape that is far from a 2-monotone. However, if Y_j passes through a "good" S-box, the data can be transformed to follow a 2-monotone distribution (in particular, a monotone distribution). For example, if $W_1 \| W_2$ passes through an S-box S in Table 3, $D_{S(W_1 \| W_2)}$ follows a monotone distribution. This is illustrated in Figure 15. The method for creating such S is simple. After obtaining the distribution of concatenated data, arrange the distribution values in ascending order and input the elements of the domain that provide the distribution values into the S-box in order. For example, as shown in Figure 14, $D_{W_1 \| W_2}$ has a minimum value at $x = 15$ and the second-smallest value at $x = 3$. If x values are arranged in this manner, they become 15, 3, 12, 7,$\cdots$, 2, 6, and 10. Inputting these in sequence into the S-box creates S results in Table 3.

Table 3. 4-bit S-box, which is used with $W_1 \| W_2$ to create monotone distribution.

x	15	3	12	7	13	11	0	1	4	14	8	5	9	2	6	10
$S(x)$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Although we provide an example of transforming 4-bit data created by concatenating two 2-bit entropy sources, this method can be applied to arbitrary n-bit data. With this method, even if the input sequences do not follow the 2-monotone distribution, Theorem 2 can be applied. However, memory is required to store the S-box, and the accumulation speed can be reduced owing to additional operations. Additionally, significant amounts of meaningful data may be required to estimate the distribution. Figure 16 illustrates the Windows RNG entropy accumulation process using an additional S-box.

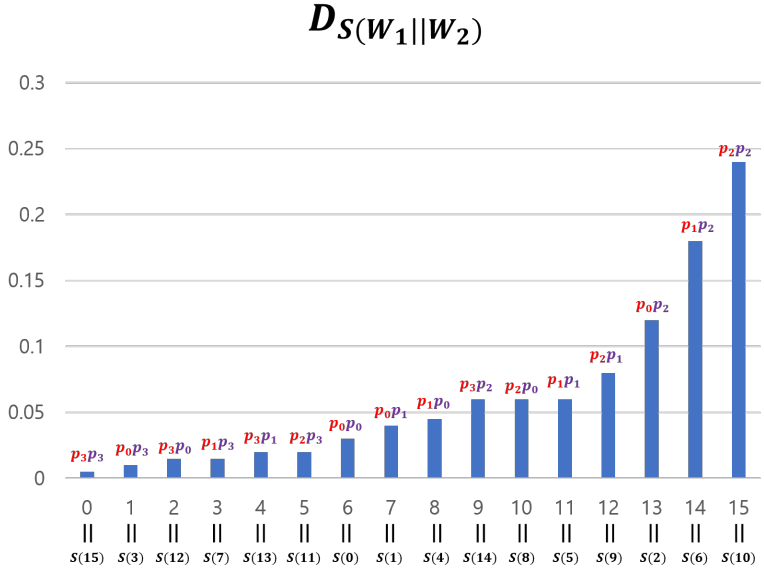


Figure 15. Distribution of S-boxed concatenated entropy sources.

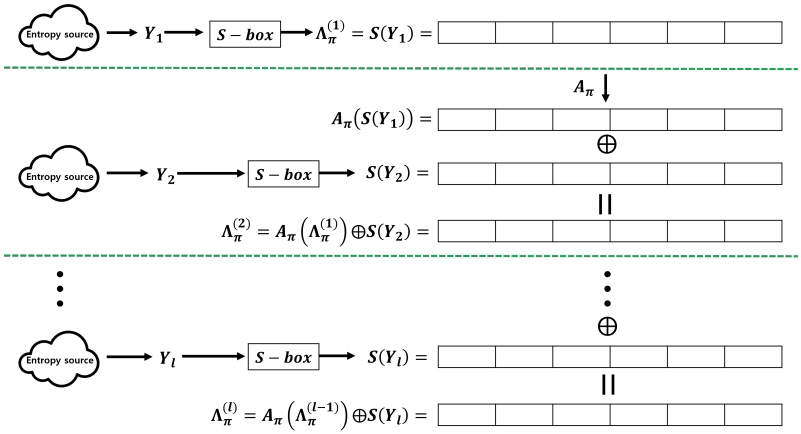


Figure 16. Windows RNG entropy accumulation with additional S-box.

4.2.3. Applying Theorem 2

In this subsection, we determine the number l required for $\Lambda_\pi^{(l)}$ to exceed a min-entropy of 7.86-bit per 8-bit by applying Theorem 2 to $S(Y_j)$. We did not conduct actual experiments because of the numerous S-boxes that needed to be implemented and the vast amount of data required to estimate D_{Y_j}.

To apply Theorem 2, the first step is obtaining $H_{min}(D_{S(Y_j)})$; as the S-box does not change the min-entropy, $H_{min}(D_{Y_j})$ is used instead. Using [17] to estimate $H_{min}(D_{Y_j})$ requires 1,000,000 pieces of Y_j, which is practically impossible. Therefore, we set $k = 4.9378$, which is the lower bound of $H_{min}(D_{Y_j})$, estimated using 2000 pieces of W_i data, as mentioned in Section 3.5. Note that the estimated min-entropy at $l = 1$ in Table 1 of Section 3.5 and the previously estimated min-entropy have explicitly different estimation targets.

With $k = 4.9378$, $k' = \left\lfloor \frac{k}{2} \right\rfloor = 2$. If we set $\pi = rot_{(2,8)}$, the covering number $m = C_{\pi,k'}$ becomes four, which is the minimum value of every possible π. Thus, we opted to use $rot_{(2,8)}$ as a bit permutation for the entropy accumulation. Considering these considerations, l can be calculated as follows:

$$\begin{aligned}
& n\left(1 - 2^{\frac{k}{2} - \frac{kl}{2m}}\right) \geq 7.86 \\
\Rightarrow\ & \frac{n - 7.86}{n} \geq 2^{\frac{k}{2}(1 - \frac{l}{m})} \\
\Rightarrow\ & \frac{\ln(n - 7.86) - \ln n}{\ln 2} \geq \frac{k}{2}\left(1 - \frac{l}{m}\right) \\
\Rightarrow\ & l \geq m\left[1 - \frac{2}{k}\left(\frac{\ln(n - 7.86) - \ln n}{\ln 2}\right)\right] \\
\Rightarrow\ & l \geq 4\left[1 - \frac{2}{4.9378}\left(\frac{\ln(8 - 7.86) - \ln 8}{\ln 2}\right)\right] = 13.4559.
\end{aligned}$$

That is, if we assume $l = 14$ to create $\Lambda_\pi^{(l)}$, $H_{min}(D_{\Lambda_\pi^{(l)}})$ will exceed the value 7.86.

In Section 3.4, we observed that our theory yields $l = 16$. Although Theorem 2 may yield slightly superior results, the difference is insignificant. Considering the time consumed for the additional S-box and bit permutations and the storage space for S-boxes, we can conclude that our entropy accumulation provides more practical results.

5. Conclusions

The contributions of our study can be summarized as follows. First, we have proposed entropy accumulation of the Fast-Refresh type, which is composed of bitwise XOR alone without hash functions, and have proved the theorem that requires only the independence without identical distribution condition of input sequences.

Second, we have established 7.86 as the lower bound for the min-entropy per 8-bit, which was considered secure based on the three benchmarks. To surpass this lower bound, our proposed theory yielded iteration number $l = 16$.

We have implemented an actual RNG to verify the theory. Our experimental results have indicated that if we use XOR operations just four times, the generated output sequences exceeded the lower bound. The entropy source used in this experiment is an image sensor PV 4209 K. This entropy source is a QRNG that utilizes dark shot noise to generate random numbers. The most important property of our entropy source is the independence of pixels. Since each piece of 2-bit data from pixels is considered as an independent random variable, we can apply the main theorem to obtain the lower bound of the min-entropy.

Finally, we have compared our entropy accumulation with two types of entropy accumulations, which are Slow-Refresh of IDQ QRNG and Fast-Refresh of Windows RNG.

As a further study, we would like to consider various entropy accumulations that have more general and practical applications than our proposed Fast-Refresh mechanism.

Author Contributions: Formal analysis, Y.C.; Writing–original draft, Y.C.; Writing—review & editing, J.-S.K. and Y.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2021M1A2A2043893) and Ministry of Science and ICT, South Korea (1711174177).

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Kwek, L.C.; Cao, L.; Luo, W.; Wang, Y.; Sun, S.; Wang, X.; Liu, A.Q. Chip-based quantum key distribution. *AAPPS Bull.* **2021**, *31*, 1–8. [CrossRef]
2. Dodis, Y.; Guo, S.; Stephens-Davidowitz, N.; Xie, Z. No time to hash: On super-efficient entropy accumulation. In Proceedings of the Advances in Cryptology—CRYPTO 2021: 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, 16–20 August 2021; Proceedings, Part IV 41; Springer: Berlin/Heidelberg, Germany, 2021; pp. 548–576.
3. Shoup, V. *A Computational Introduction to Number Theory and Algebra*; Cambridge University Press: Cambridge, UK, 2009.
4. Dodis, Y. *Randomness in Cryptography*; Spring: Berlin/Heidelberg, Germany, 2013.
5. Hayashi, M.; Tsurumaru, T. More efficient privacy amplification with less random seeds via dual universal hash function. *IEEE Trans. Inf. Theory* **2016**, *62*, 2213–2232. [CrossRef]
6. Hayashi, M. Exponential decreasing rate of leaked information in universal random privacy amplification. *IEEE Trans. Inf. Theory* **2011**, *57*, 3989–4001. [CrossRef]
7. Ferguson, N. *The Windows 10 Random Number Generation Infrastructure*; Microsoft Corporation: Redmond, WA, USA, 2019.
8. Herrero-Collantes, M.; Garcia-Escartin, J.C. Quantum random number generators. *Rev. Mod. Phys.* **2017**, *89*, 015004. [CrossRef]
9. Lin, X.; Wang, R.; Wang, S.; Yin, Z.Q.; Chen, W.; Guo, G.C.; Han, Z.F. Certified randomness from untrusted sources and uncharacterized measurements. *Phys. Rev. Lett.* **2022**, *129*, 050506. [CrossRef] [PubMed]
10. Liu, W.B.; Lu, Y.S.; Fu, Y.; Huang, S.C.; Yin, Z.J.; Jiang, K.; Yin, H.L.; Chen, Z.B. Source-independent quantum random number generator against tailored detector blinding attacks. *Opt. Express* **2023**, *31*, 11292–11307. [CrossRef] [PubMed]
11. Zhou, H.; Li, J.; Zhang, W.; Long, G.L. Quantum random-number generator based on tunneling effects in a Si diode. *Phys. Rev. Appl.* **2019**, *11*, 034060. [CrossRef]
12. Zhou, Q.; Valivarthi, R.; John, C.; Tittel, W. Practical quantum random-number generation based on sampling vacuum fluctuations. *Quantum Eng.* **2019**, *1*, e8. [CrossRef]
13. Park, B.K.; Park, H.; Kim, Y.S.; Kang, J.S.; Yeom, Y.; Ye, C.; Moon, S.; Han, S.W. Practical true random number generator using CMOS image sensor dark noise. *IEEE Access* **2019**, *7*, 91407–91413. [CrossRef]
14. Tawfeeq, S.K. A random number generator based on single-photon avalanche photodiode dark counts. *J. Light. Technol.* **2009**, *27*, 5665–5667. [CrossRef]
15. Wang, F.X.; Wang, C.; Chen, W.; Wang, S.; Lv, F.S.; He, D.Y.; Yin, Z.Q.; Li, H.W.; Guo, G.C.; Han, Z.F. Robust quantum random number generator based on avalanche photodiodes. *J. Light. Technol.* **2015**, *33*, 3319–3326. [CrossRef]
16. Matsui, M. Linear cryptanalysis method for DES cipher. In Proceedings of the Advances in Cryptology—EUROCRYPT'93: Workshop on the Theory and Application of Cryptographic Techniques, Lofthus, Norway, 23–27 May 1993; Proceedings 12; Springer: Berlin/Heidelberg, Germany, 1994; pp. 386–397.
17. Turan, M.S.; Barker, E.; Kelsey, J.; McKay, K.A.; Baish, M.L.; Boyle, M.E. Recommendation for the entropy sources used for random bit generation. *NIST Spec. Publ.* **2018**, *800*, 102.
18. Peter, M.; Schindler, W. *A Proposal for Functionally Classes for Random Number Generators*; AIS 20/31; BSI: Hamburg, Germany, 2022.
19. Troyer, M.; Renner, R. A randomness extractor for the Quantis device. *Quantum Number Gener.* **2001** , 2001–2010.
20. IDQ. *Randomness Extraction for the Quantis True Random Number Generator*; IDQ: Geneva, Switzerland, 2012.