Ryu *et al. J Surveill Secur Saf* 2024;5:234-57
DOI: 10.20517/jsss.2024.16

Journal of Surveillance, Security and Safety

Original Article

Open Access

Entropy harvest and key derivation from the image sensors in IP camera

Jieun Ryu[1], Gwangjae Kim[1], Jeongbeen Ko[1], Dongkeun Kang[2], Ju-Sung Kang[1,3], Yongjin Yeom[1,3]

[1]Department of Financial Information Security, Kookmin University, Seoul 02707, Republic of Korea.
[2]SDT Inc., 10F, 5, Teheran-ro 44-gil, Gangnam-gu, Seoul 06211, Republic of Korea.
[3]Department of Information Security, Cryptography and Mathematics, Kookmin University, Seoul 02707, Republic of Korea.

Correspondence to: Prof. Yongjin Yeom, Department of Information Security, Cryptography and Mathematics, Kookmin University, 77 Jeongneung-ro, Seongbuk-gu, Seoul 02707, Republic of Korea. E-mail: salt@kookmin.ac.kr; ORCID: 0000-0002-8240-8661

How to cite this article: Ryu J, Kim G, Ko J, Kang D, Kang JS, Yeom Y. Entropy harvest and key derivation from the image sensors in IP camera. *J Surveill Secur Saf* 2024;5:234-57. http://dx.doi.org/10.20517/jsss.2024.16

Received: 1 Jul 2024 **First Decision:** 21 Oct 2024 **Revised:** 11 Nov 2024 **Accepted:** 12 Dec 2024 **Published:** 31 Dec 2024

Academic Editor: Qiong Huang **Copy Editor:** Ting-Ting Hu **Production Editor:** Ting-Ting Hu

Abstract

With the widespread use of connected surveillance systems using IP cameras, the security of video data has become a critical issue. In response to the potential compromise of sensitive information via covert channels, the United States and the United Kingdom warned against the use of suspicious cameras in critical infrastructure. To mitigate the security concerns, it is indispensable to investigate every component inside IP cameras using evaluation and validation programs such as Common Criteria and Cryptographic Module Validation Program. However, such compliance tests on implementation under test cannot suffice to prevent the products from malicious attacks using a Trojan-horse in a black-box component, particularly inserted in the image-sensing module or the random number generator (RNG).

In this paper, we categorize potential vulnerabilities and argue that IP cameras should disclose the components of the modules to mitigate the threats of backdoor. Based on this, we provide a way to construct backdoor-free RNG. To remove the possibility of the backdoor inside the RNG, we extract randomness from optical black pixels of the image sensor and apply two entropy accumulation methods to guarantee the min-entropy of accumulated data under weak assumptions, which means the IP camera already has a hardware entropy source in itself without additional suspicious entropy source. In the experiment using an IP camera with a Cortex-A53 processor, we can harvest entropy at the rate of 7.3kbps. The operating speed of the RNG is sufficient to provide random bits without delay when a symmetric key is updated every minute.

Keywords: Entropy accumulation, image sensor, parallel noise source, random bit generation, key derivation

1. INTRODUCTION

As the demand for real-time video data continues to grow and the interconnectivity of cameras and smart devices increases, the major focus for the real-time surveillance camera market is shifting from traditional closed-circuit cameras to IP cameras. This change is driven by the increasing security threats, infrastructure trends in the Asia-Pacific area, commercial security considerations in the Middle East, and the aim to reduce the national crime rate[1,2]. Furthermore, the demand from individuals and governments for security cameras is also driving the growth of the market[3]. The size of the global IP camera market was estimated to be $9.93 billion in 2021, and is expected to reach $24 billion by 2028[4–6]. In this way, the IP camera market has grown rapidly over the past decade, and is expected to maintain a compound annual growth rate (CAGR) of more than 13%.

However, the growing use of remote-controlled cameras such as IP cameras, has led to the discovery of a number of vulnerabilities and deliberate misuses, raising security concerns about the negative aspects of these devices. In South Korea, a study by the Ministry of Science and ICT and the Korea Internet & Security Agency (KISA) of over 400 commercially available IP cameras uncovered security vulnerabilities in 126 products, including weak initial identification or password (ID/PW) settings. In response to these security vulnerabilities, the South Korean government has strengthened security requirements for all domestically distributed IP cameras since February 2019[7]. Nonetheless, there are still concerns about security vulnerabilities and potential security threats, particularly in products from foreign manufacturers. For instance, a backdoor found in Dahua Technology's camera in 2017 could be activated despite resetting the initial password. Despite the recall of the problematic cameras, two years later in 2019, a new backdoor program was discovered again, raising issues of potential eavesdropping even when the camera is off[8–10]. In addition, security issues occur without ceasing in private use, such as the remote control issues of IP camera devices and relevant programs that occurred in South Carolina in 2018 and the hacking of Ring cameras in 2019[11,12]. The lesson learned from the above cases is that the security of IP cameras is not limited to secure transmission of video data but also considers secondary damage such as distributed denial of service (DDoS) attacks initiated by infected IP cameras as network devices or malicious control of IP cameras hacked by Trojan horses.

In particular, China accounts for 45% of IP camera market sales, and has the largest IP camera market share, with a growth rate of 13.5%, compared to about 5% for the rest of the world[13–15]. However, vulnerabilities suspiciously exist not only in products from Dahua Technology but also in those from other Chinese companies[16]. In 2018, the United States removed them from an army base in Missouri due to potential security concerns. In 2021, the Federal Communications Commission (FCC) announced concerns that equipment manufactured by suspicious companies could threaten the security of telecommunications networks in the U.S.[17–19]. Accordingly, a new regulation was announced recommending the cancellation of approval of vulnerable video security products. The British government also decided to stop using Chinese CCTV such as Hikvision in 2022[20]. This comes from various security concerns after China implemented the National Intelligence Act in 2017, which strengthens its authority to surveillance and investigate dissidents. Then, in 2021, several security issues were raised due to the leak of the private life of British aide Matt Hancock to the media through a Hikvision camera[21]. As such, security concerns are increasing in each country due to the rapid expansion of remote cameras from overseas manufacturers not only about hacking, but also about the possibility of data leakage, illegal surveillance, and other secondary damage through hidden covert channels.

In order to enhance the security of IP camera systems, in 2024, South Korea's National Innovation System (NIS) issued the 'Security Conformance Verification Policy for Product Family of Image Information Processing Device in Public Sector', stating that verification must be conducted for preliminarily removing the hidden backdoors and known vulnerabilities in image processing devices and addressing the vulnerabilities[22]. This policy calls on internal core modules of intelligent CCTV, such as IP cameras, to verify security conformance and requires that each component be verified. In general, IP camera manufacturers collect internal parts from

other vendors, assemble them into a single product, and then apply encryption to the camera by combining a network module and a cryptographic module. In the case of the image sensor module for cameras, black-box modules with ethernet ports are widely used, but it is difficult to find and eliminate possible vulnerabilities because their design and structure are not open to the public. Therefore, it is necessary to investigate thoroughly each component of the camera including the image sensor module, cryptographic module, and network module from the initial stage of production. However, such verification may increase the production costs, also when a separate module or dedicated hardware noise source is used to provide the random bit through the random number generator (RNG), which is essential for cryptographic modules. Furthermore, the addition of hardware noise sources significantly influences the module's area occupation. Nevertheless, it is important to provide sufficient entropy to the cryptographic module in IP cameras.

1.1. Our contributions

We argue that, to prevent the possibility of backdoors in an IP camera, the camera's RNG and encryption modules should be designed in compliance with security standards, and architecture of all modules inside the IP camera should be investigated for applying validation. Furthermore, using an image sensor as a noise source enables us to build cryptographically strong RNG without additional hardware where a backdoor could exist. In this paper, we categorize the potential vulnerabilities and provide a way to construct secure IP cameras by adopting the open architecture of the modules to eliminate the possibility of backdoors inside the camera. We propose an efficient method for designing a backdoor-free RNG that does not require additional hardware noise sources. Then, we present the cryptographic procedures and experimental results supporting the feasibility of the proposed approach. This enables key generation without the possibility of malicious intervention by an adversary. Specifically, we propose an efficient random number generation technique based on image sensors and provide a direction to fundamentally resolve concerns about known security vulnerabilities of IP cameras from the image generation stage to the video encryption stage.

- We specifically present IP camera covert channel vulnerabilities and directions for preventing that channel in the cryptographic module by ensuring the reliability of the entire process of camera image creation, key generation, and data encryption.
- We present a method to ensure sufficient entropy to operate an RNG for generating the encryption keys and sensitive security parameters in a validated encryption module.
- By implementing our encryption scheme in a commercial full high definition (FHD) camera, we achieve the min-entropy of 0.99 per bit at an average rate of 7.3kbps, which is enough to encrypt 15 frames per second.

The rest of this paper is structured as follows. In Section "RANDOM NUMBER GENERATION IN IP CAMERA", we first analyze the security threat model that can occur in IP cameras and show how to mitigate each threat. Next, we explain the noise source, which is a source of randomness that constitutes a true RNG (TRNG), introduce the entropy pool using RNG of Linux and Windows, and then explain how to evaluate the entropy of the noise source. In Section "ENTROPY HARVEST", we introduce two methods for efficiently accumulating entropy from noise sources. These methods guarantee a theoretical lower bound of accumulated entropy using only simple operations, enabling high-speed entropy harvesting. This compensates for the limitation of raw random number generation speed from dark shot noise of image sensors. Additionally, we propose a health test technique that utilizes the parallel noise sources of an image sensor, and a method of configuring and operating an entropy pool through the accumulation technique. Unlike the general entropy accumulation techniques described in Section "RANDOM NUMBER GENERATION IN IP CAMERA", the proposed techniques ensure a theoretical lower bound per bit of entropy and are designed to minimize time-memory costs when operating an entropy pool. Finally, in Section "KEY DERIVATION AND EXPERIMENTAL RESULTS", we describe the overall construction for encrypting data from IP cameras along with the TRNG discussed in Section "ENTROPY HARVEST". We also describe the algorithms used for each component and evaluate the performance of our model. In the algorithm selection process, we prioritized lightweight implementation con-

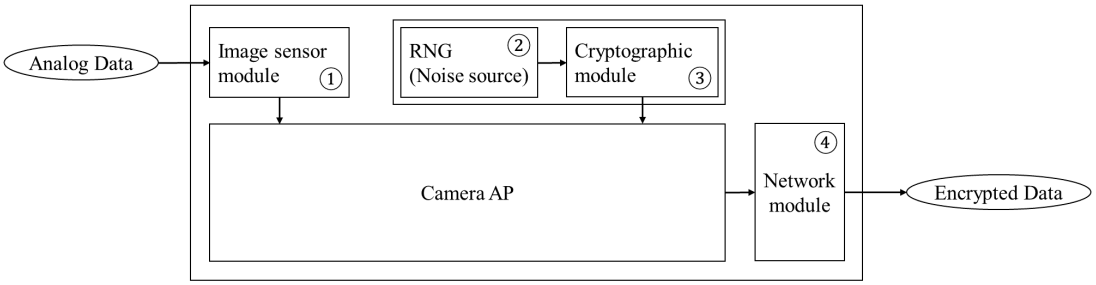


Figure 1. Structure of IP camera.

sidering that IP cameras are used in an Internet-of-Things (IoT) environment, and confirmed whether the high-speed implementation might be expected in Section "ENTROPY HARVEST" through experiments.

2. RANDOM NUMBER GENERATION IN IP CAMERA

2.1. Security threats to IP cameras

Structure of IP camera

The basic configuration of an IP camera for communication with encrypted video data is shown in Figure 1. In this figure, the image sensor module receives analog video data and digitizes it, and the camera application performs video data processing and overall camera control. The cryptographic module provides random bits for key derivation and encrypts digitized data. In addition, the RNG of the cryptographic module may use an external hardware randomness source. The network module performs the process of transmitting encrypted video data over the network.

The encryption process of the secure IP camera is as follows. When a user establishes IP communication with a camera through a device, the user device and the camera have to share a symmetric key for encrypting and decrypting video data. In general, the encryption key is shared using key agreement protocol or public key cryptography. The data can be encrypted with a symmetric encryption algorithm using their shared key. The following steps show a typical process using public key cryptography.

Step 1) Public key generation for user device: The user's device generates a private key sk based on the user ID and PW or the device information, and a corresponding public key pk.
Step 2) Symmetric key generation on IP camera: The camera generates a secret symmetric key k using RNG.
Step 3) Encrypting and sending the symmetric key: When communication is established, the IP camera encrypts the symmetric key k using the public key pk, and sends the encrypted key $E_{pk}(k)$ to the recipient (user device).
Step 4) Symmetric key decryption: The user device decrypts $E_{pk}(k)$ into k using the private key sk and stores it.
Step 5) Video data encryption: The IP camera encrypts video data P into C using the symmetric key k and sends it to the user device.
Step 6) Video data decryption: The user device decrypts received data C into P using the symmetric key k to obtain video data.

The symmetric key can be updated whenever the user device establishes communication. In this case, we can consider several methods to share a new symmetric key k' and encrypt the video data. One method is that the IP camera encrypts the new symmetric key k' with pk and sends $E_{pk}(k')$ to the user device, and then encrypts the video data with k'. Instead of sending them separately, the encrypted key $E_{pk}(k')$ and video data C can be concatenated as $E_{pk}(k')\|C$.

Table 1. Classification of hidden channels in IP cameras

Location	Image sensor module	Network module	Cryptographic module
Channel	Covert Channel		Subliminal channel
Attack Technique	Steganography	various attacks	Kleptography
Place	①	④	②, ③

Security threat model for IP cameras

Regardless of the encryption process adopted, the IP cameras still have vulnerable points. Figure 1 shows these points as numbers 1, 2, 3, and 4. Unfortunately, these vulnerabilities come from network functionality, which is the main aim of the IP camera. For example, some malicious attackers may insert their IP addresses as another destination of IP communication and receive the camera data at the same time as legitimate users. The IP addresses for this attack are generally concealed in the network-enabled camera module. Also, secret messages such as the location of the camera can be embedded using steganography. Therefore, IP cameras with black-box components have potential vulnerabilities.

Moreover, if the module associated with the encryption process is also kept as a black box, it may contain hidden paths such as the covert or subliminal channel. Butler Lampson and Gustavus Simmons showed that even in the cryptographic communication, it is possible for certain communicating parties to establish a hidden channel among them without revealing information to anyone else. In addition to confidentiality, availability, non-repudiation, and integrity, they noted that there are additional conditions that a cryptosystem must satisfy[23]. The hidden channel that can appear in an IP camera whose internal structure is unknown to the public is called the covert channel or the subliminal channel. The subliminal channel is a kind of covert channel that uses cryptographic techniques.

Covert channel, introduced by Lampson in 1973, is a channel "not intended for information transfer at all, such as the service program's effect on system load", which is distinguished from a legitimate channel that controls the communication access according to computer security policies[24]. An example of an attack using this channel is steganography in which the attacker hides a secret message in an original message so that other participants are unaware of the existence of the hidden channel. Therefore, it is possible for a malicious camera manufacturer to create covert channels on the image sensor module and network module to perform a steganography attack[25–27].

The subliminal channel, proposed by Simmons in 1984, differs from a typical covert channel in that "even if the monitor knows what to look for, he can't discover either the message or the usage of the channel"[28]. This channel guarantees computational concealment based on cryptographic techniques. When a backdoor is created or when a hidden message is sent through the channel, the channel's existence is less likely to be detected or the hidden message revealed compared to a typical covert channel. The kleptographic attack is an exquisite example of the attack using a subliminal channel, which creates an asymmetric backdoor. In the SETUP attack, using the backdoor in the RNG related to the key generation, the attacker can extract the private key from the corresponding public key[29]. This attack is hardly detected by the security evaluation of black-box modules without a thorough investigation of the internal components.

In the aforementioned structure, as shown in Figure 1, backdoors can be found in several components. Table 1 summarizes the hidden channels established by backdoors that can be found in IP cameras.

To prove the absence of the backdoors, architectures and components should be investigated thoroughly by the validation programs. This would enable users to conduct thorough security assessments, including validation through recognized programs such as the Cryptographic Module Validation Program (CMVP) or the Common Criteria (CC). Such an approach would significantly contribute to mitigating the potential threats of IP cameras.

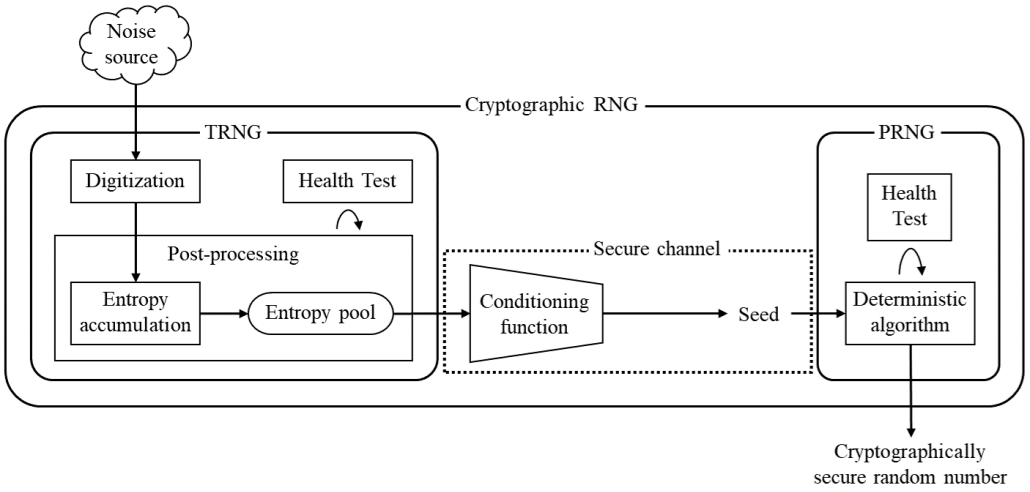


Figure 2. Structure of Cryptographic RNG.

2.2. Random bit generation

As shown in Figure 2, the structure of RNG can be divided into two parts: entropy extraction and pseudorandom bit generation. Unpredictable random numbers originate from uncertainty in noise sources. The entropy extraction part harvests entropy from the noise sources and accumulates it into the entropy pool. The entropy pool is managed to hold entropy as high as possible. The pseudorandom bit generation part takes an input, called a seed, then generates a bit sequence that is statistically hard to distinguish from random bits and fair coin tossing. Since output bits are produced deterministically according to the standard algorithm, the same seed results in the same output. Therefore, it is important to use unpredictable, high-entropy seeds. The final output of RNG, used as encryption keys, nonces, initialization vectors (IVs), and so on, becomes vulnerable if an attacker predicts the seed successfully. Once again, we emphasize that randomness heavily relies on the noise source.

Furthermore, the RNG contains a health test that does not operate as part of the main random number generation process. A health test works according to a specific period or based on user commands. This test does not influence the entropy of the RNG's output but is used to detect failures in the entropy source.

2.3. Source of randomness

As specified in SP 800-22 of the National Institute of Standards and Technology (NIST), which analyzes the statistical properties of pseudorandom number generator (PRNG) output, if the rate of change of the noise used in seed construction is low, the generated seed can be a predictable random number even if the randomness of PRNG outputs satisfies the criteria proposed in the document [30]. In other words, the entropy of the noise source fundamentally affects the security of the random number, and the user must increase the entropy of the seed to ensure that the output of the RNG satisfies the unpredictability. To this end, the designer of the RNG should select and use noise sources that can provide sufficient entropy.

The noise sources used in a cryptographic RNG are categorized into hardware and software noise sources. Hardware noise sources include thermal noise, which utilizes non-deterministic phenomena in electronic circuits, free running ring oscillators, and radiative decay and photoelectric effects, which use physical phenomena. Software noise sources include interrupts, changes in memory values or area sizes, program event responses, file information, network status, and values from input devices such as mouse and keyboard. The operating system (OS) uses software noise sources such as users' input from the keyboard, mouse, camera, touch screen and disk state, and interrupts as noise sources without adding any external hardware. Addition-

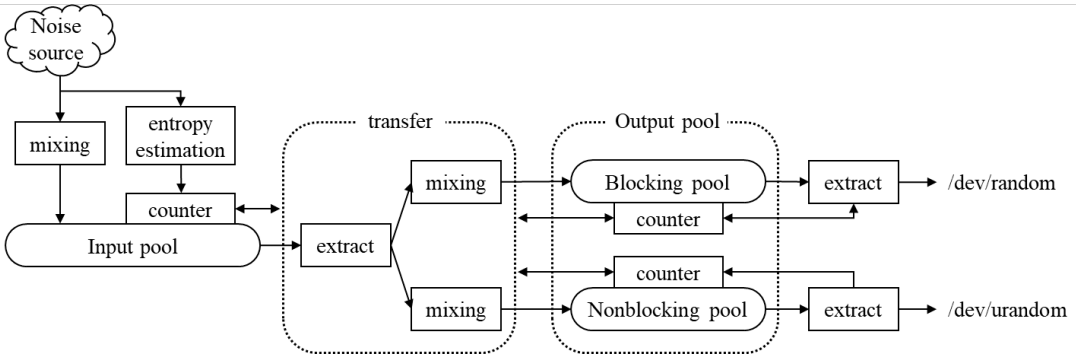


Figure 3. Structure of the Linux kernel RNG.

ally, the hardware noise source built into the CPU chip can be used as a hardware noise source[31].

When multiple noise sources are used, the seed is formed by combining the outputs of each noise source. Using two or more noise sources relieves the entropy degradation caused by the bias of the noise source, and allows the noise sources to be replaced without stopping the entire RNG when an error occurs in a certain noise source. NIST SP 800-90B (Recommendation for the Entropy Sources Used for Random Bit Generation) can be referred to evaluate each noise source and to estimate the level of entropy.

Dark shot noise

Without any additional hardware noise sources, IP cameras can utilize dark shot noise as a physical noise source because each optical black pixel (OBP) in the image sensor can be regarded as an independent noise source[32]. A shot noise-based quantum RNG (QRNG) proposed in 2014 by Sanguinetti *et al.* extracts raw random data in parallel from multiple pixels in a single image sensor[33]. According to the noise source classification of RNGs presented at the NIST International Cryptographic Module Conference in 2021, the shot noise from image sensors is classified as a quantum random number, and image sensors can be used to generate quantum random numbers by taking photon shot noise or dark shot noise. While a photon shot noise-based RNG requires a dedicated light source, a dark shot noise-based RNG uses pixels on the boundary of the image sensor. However, the speed of the random number generation of dark shot noise is relatively slower due to the smaller number of pixels and noise bits.

2.4. Entropy pool

The entropy pool, a concept introduced by Kelsey *et al.* in 1999 as a way to prevent Iterative Guessing Attacks, is a buffer or memory that stores the outputs obtained from the sources of randomness[34]. Each output of the noise source is mixed with previously stored values in the pool to increase the entropy of the pool. Whenever the pool provides random bits, its entropy decreases. The entropy pool can be managed by the OS as in the Linux RNG (LRNG) and the Windows RNG (WRNG).

Entropy pools in LRNG

The LRNG has three types of entropy pools: input, blocking, and non-blocking pools, as shown in Figure 3[35]. The input pool is a 512-byte pool that accumulates entropy to deliver high entropy when the noise source data is requested from the blocking or non-blocking pool. The blocking pool and non-blocking pool are both 128-byte output pools. When data is accumulated in the pool through the mixing function, the entropy of the pool increases and the entropy counter goes up. Conversely, when a random number is extracted, the entropy of the pool decreases and the counter goes down. The blocking pool would not produce any output when it does not hold enough entropy.

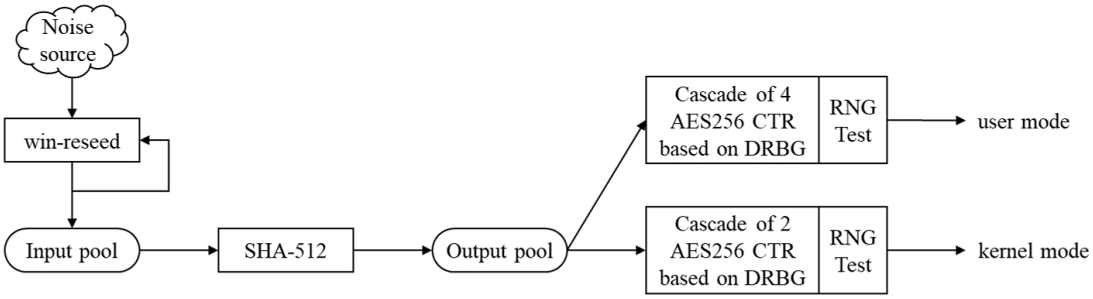


Figure 4. Structure of the Windows 10 RNG.

As shown in Figure 3, all three pools have counters that indicate the amount of entropy in the pool. With these entropy counters, which interoperate with the entropy estimation function and extract function, LRNG roughly and empirically estimates the entropy. We describe the details of the entropy evaluation further.

Entropy pools in WRNG

The entropy pools of WRNG consist of an input pool and an output pool as shown in Figure 4. The input pool accumulates data collected from various noise sources, and the output pool stores data with high entropy for output generation[36]. The data from noise sources are accumulated in the input pool through a win-reseed function that consists only of bit shift and exclusive-or (XOR) operation. In 2020, Dodis *et al.* proved that this function theoretically guarantees the lower bound of min-entropy[37]. The hash function SHA-512 is used to transfer data from the input pool to the output pool and the cascade of deterministic random bit generators (DRBG) using the Advanced Encryption Standard (AES) block cipher generates random bits from the output pools.

Unlike the LRNG, which manages entropy using a mixing function and counter with a hash function to ensure the randomness of output random numbers, the WRNG uses simple bitwise operations for the input pool to accumulate entropy, and a hash function for the output pool. After accumulating entropy, the randomness of the output is guaranteed by RNG test, which determines whether the data is sufficiently random at the final step. WRNG does not use counters in either pool. Instead, it performs the RNG test before generating a random number for the user's request.

2.5. Entropy estimation

As previously explained, the source of randomness that provides the primary entropy of the RNG has a significant impact on the security of random numbers. Thus, it is important to estimate the entropy of each noise source. We can refer to NIST SP 800-90B and ISO/IEC 20543 for entropy estimation.

NIST SP 800-90B

SP 800-90B is a standard document for entropy source design and evaluation published by NIST in 2018. It measures the min-entropy of a source of randomness by applying conservative threshold[38]. The entropy measurement process of SP 800-90B consists of checking whether the data is IID or not, estimating the entropy, reducing the overestimation of the entropy through a restart test, and estimating the entropy. NIST provides the estimation tool consisting of their track.

- Determine the track: The entropy estimation track depends on the type of data: IID or not. We set the null hypothesis (H_0) as 'data is IID' and try to find counterevidence for it. In SP 800-90B, the process of finding counterevidence is called the Permutation test which consists of 11 tests. Furthermore, there are five Chi-square statistical tests to check for additional statistical problems.
- IID track estimate: For IID data, the entropy is estimated using the Most Common Value Estimate. The

estimation calculates the maximum probability with the most frequently occurring sample, and calculates min-entropy as the upper bound using the 99% confidence interval estimation method.
- Non-IID track estimate: For non-IID data, the entropy is estimated by ten tests for non-IID track. Then, the smallest value among these is selected as min-entropy of the data.
- Restart test: The purpose of the restart test is to prevent overestimation of entropy. This test constructs a matrix with samples obtained from the RNG using the target noise source, calculates the minimum entropy for the rows and columns, and selects the minimum of them as min-entropy of the data.
- Conditioning component: The optional conditioning component uses a deterministic function to reduce the bias and increase the entropy rate of output.

ISO/IEC 20543

ISO/IEC 20543 (Test and analysis methods for random bit generators with ISO/IEC 19790 and ISO/IEC 15408) is a standard document published in 2019 [39]. It provides guidance on statistical testing and entropy evaluation referring to NIST's SP 800-90B, SP 800-22, and the Federal Office for Information Security (BSI)'s AIS.31. For this purpose, RNG should be designed based on a stochastic model and heuristic analysis of the noise source.

3. ENTROPY HARVEST

A cryptographic RNG sequentially performs randomness extraction from noise source, digitization, entropy accumulation, and pseudorandom number generation. In the entropy accumulation phase, the digitized data collected from the noise source is accumulated to produce data with high entropy, which can be used as a seed for a PRNG. With this seed as the input of a deterministic algorithm, the PRNG generates the final random bits for various applications. This section describes systematic methods for accumulating entropy by using the image sensor of an IP camera as a parallel noise source.

3.1. Notation

The notations used in this chapter are defined as follows:

- $\pi : \{0, 1, \cdots, n-1\} \rightarrow \{0, 1, \cdots, n-1\}$: n-bit permutation defined as a bijection on a set of n elements.
- $\{0,1\}^n$: n-bit string, also denoted by $\{(b_1, b_2, \cdots, b_n) : b_1, \cdots, b_n \in \{0,1\}\}$.
- A_π : the linear transformation over $\{0,1\}^n$ corresponding to the permutation π. That is, $A_\pi(x_0, x_1, \ldots, x_{n-1}) = \left(x_{\pi(0)}, x_{\pi(1)}, \ldots, x_{\pi(n-1)}\right)$.
- D_X : The probability distribution of a random variable X.
- $H_\infty(D_X)$: the min-entropy of the probability distribution D_X defined by $H_\infty(D_X) = -\log_2 \|D_X\|_\infty$, where $\|D_X\|_\infty$ denotes the maximum probability when D_X is a discrete distribution with finite support.
- $\Lambda_\pi^{(l)} = \sum_{i=1}^{l} A_\pi^{l-i}(Y_i)$ for n-bit random variable $Y_i \in \{0,1\}^n$.
- $\|D_X\|_{\min}$ is defined for a probability mass function $f : \mathbb{X}_X \rightarrow \{0,1\}$ of a discrete random variable X as $\|D_X\|_{\min} := \min_{x \in \mathbb{X}_X}\{f(x)\}$.
- $\Gamma^{(l)} = \sum_{j=1}^{l} Y_j$, where n-bit random variables $Y_i \in \{0,1\}^n$.

3.2. Entropy accumulation

As mentioned in the previous section, the dark shot noise of an IP camera image sensor can be utilized as a parallel source of randomness and entropy can be extracted from multiple OBPs in a single image sensor. Suppose there are l OBPs in the image sensor. Let $Y_1, Y_2, \ldots, Y_l$ be random variables that represent n-bit binary data generated by the dark shot noise of each OBP in the image sensor. The distribution of each Y_i is expected to be similar, but not necessarily identical. However, the dark shot noises from each pixel do not influence one another. So, $Y_1, Y_2, \ldots, Y_l$ can be assumed independent. As initial raw data, Y_i extracted from the dark shot noise has relatively low entropy. Therefore, an entropy-enhancing process is required.

The process of entropy accumulation can be categorized into *Slow-refresh* and *Fast-refresh* based on underlying

operations[37]. Slow-refresh, widely adopted in most RNGs, involves a traditional approach utilizing hash functions. In this method, the lower bound of min-entropy for the output sequence is guaranteed by the Leftover Hash Lemma[40]. In general, it works very slowly, though it generates a significant amount of entropy at once. Moreover, it is necessary to construct a universal hash family for the Leftover Hash Lemma and select a random element from the uniformly distributed hash family for each entropy accumulation. Consequently, the implementation of a Slow-refresh entropy accumulation algorithm based on the Leftover Hash Lemma is inefficient in practice.

On the other hand, Fast-refresh accumulates entropy using bitwise operations instead of heavy hash functions, enabling rapid generation of random numbers. Though the general security of Fast-refresh has not been fully established, Dodis *et al.* have provided a lower bound on the min-entropy per bit for the sequences of random numbers generated under specific conditions[37].

In this section, we focus on the entropy accumulation using two Fast-refresh methods, WRNG and XOR entropy accumulation to generate sequences of random numbers with high entropy from inputs $Y_1, Y_2, \ldots, Y_l$. Both entropy accumulation mechanisms ensure a theoretical lower bound on accumulated entropy when the noise source data satisfy independence conditions. Notably, even if the distributions of individual noise sources are independent but not identically distributed, these mechanisms ensure robust entropy accumulation. Furthermore, these approaches perform entropy accumulation using only simple bitwise operations. Therefore, these two mechanisms are suitable for IP cameras that utilize parallel noise sources to extract data in lightweight environments.

Note that both entropy accumulation methods rely on the condition that input data $Y_1, Y_2, \ldots, Y_l$ should satisfy a certain property. Observing the property of $Y_1, Y_2, \ldots, Y_l$ from the dark shot noise, one can choose one of the two entropy accumulation methods depending on the distributions.

Fast entropy accumulation in WRNG[36]

In Windows 10, the entropy accumulation mechanism of WRNG uses bit permutations and bitwise XOR operations iteratively on digitized noise source $Y_1, Y_2, \ldots, Y_l$. The security of the entropy accumulation in WRNG has not been fully explained yet, but Dodis *et al.* demonstrated that if the distribution of noise source satisfies certain conditions, the entropy accumulation through appropriate bit permutations guarantees the lower bound on min-entropy[37]. To explain this precisely, they introduce two definitions: 2-monotone distribution and covering number.

Definition 3.1 (Covering number) *For a permutation* $\pi : \{0, 1, \ldots, n-1\} \to \{0, 1, \ldots, n-1\}$ *and an integer* t *satisfying* $1 \le t \le n$*, the covering number* $C_{\pi,t}$ *is the smallest integer* m *that satisfies the following:*

$$\{\pi^j(i) : 0 \le i < t,\ 0 \le j < m\} = \{0, 1, \cdots, n-1\}.$$

If such an integer m *does not exist, then* $C_{\pi,t} := \infty$.

The covering number of a permutation refers to the minimum number of repetitions needed to output all possible elements of the permutation at least once. Therefore, the efficiency of the permutation increases as the covering number of the permutation decreases.

Definition 3.2 (two-monotone distribution) *A probability distribution* D_X *of an* n*-bit random variable* X *over* $\mathbb{Z}_{2^n}$ *follows a two-monotone distribution if its probability mass function* p *on* $\{0, \ldots, 2^n - 1\}$ *is a monotone function on both intervals* $[0, i]$ *and* $[i+1, 2^n - 1]$ *for some* $0 \le i \le 2^n - 1$.

That is, a probability distribution D is defined as a two-monotone distribution if it has at most one local extremum point.

One of the main findings of Dodis *et al.*[37] is that entropy can be accumulated for independent n-bit random variables $Y_1, Y_2, \ldots, Y_l$ through bitwise XOR operations and bit permutations. Specifically, given that the probability distributions $D_{Y_1}, D_{Y_2}, \ldots, D_{Y_l}$ of $Y_1, Y_2, \ldots, Y_l$ have a min-entropy of at least k and are two-monotone distributions, the number l of iterations required to ensure a lower bound h on the min-entropy per bit of output is directly proportional to h and k, and inversely proportional to the covering number $C_{\pi,t}$ of bit permutation π. Theorem 3.1 states this precisely.

Theorem 3.1 [37] *Suppose that the n-bit random variables $Y_1, Y_2, \ldots, Y_l$ satisfy the following three conditions:*

i) The distributions $D_{Y_1}, D_{Y_2}, \ldots, D_{Y_l}$ are two-monotone distributions.
ii) The distributions $D_{Y_1}, D_{Y_2}, \ldots, D_{Y_l}$ have a min-entropy of at least k ($k \geq 2$).
iii) $Y_1, Y_2, \ldots, Y_l$ are independent.

For a bit permutation $\pi : \{0, 1, \ldots, n-1\} \rightarrow \{0, 1, \ldots, n-1\}$ and $k' = \left\lfloor \frac{k}{2} \right\rfloor$, define the covering number m as $m = C_{\pi,k'}$ and the linear transformation corresponding to the bit permutation π as A_π. Then, if we define $\Lambda_\pi^{(l)} := A_\pi^{l-1}(Y_1) \oplus A_\pi^{l-2}(Y_2) \oplus \ldots \oplus Y_l$, the following holds for $l \geq m$:

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

The following Algorithm 1 is based on WRNG entropy accumulation mechanism and utilizes a sequence of n-bit input data $Y_1, Y_2, \ldots$, which satisfy the three conditions of Theorem 3.1, to accumulate entropy. Given an algorithm, it is guaranteed that the accumulated R n-bit data $(X_1, X_2, \ldots, X_R)$ have an entropy at least h per bit.

Algorithm 1 WRNG entropy accumulation [37]

Input: n-bit input data $Y_1, Y_2, \ldots$, output entropy per bit h, number of n-bit output data R, permutation π of covering number m, min-entropy k of $D_{Y_1}, D_{Y_2}, \cdots, D_{Y_l}$
Output: accumulated entropy $\Lambda_\pi^{(l)}, \Lambda_\pi^{(2l)}, \cdots, \Lambda_\pi^{(Rl)}$

1: Calculate $l = \left\lceil m \left\lceil 1 - \frac{2}{k}\left(\frac{\ln(1-h)}{\ln 2}\right)\right\rceil\right\rceil$.
2: **for** $i = 1$ to R **do**
3: $\quad \Lambda_\pi^{(il)} = A_\pi^{l-1}(Y_{(i-1)l+1}) \oplus A_\pi^{l-2}(Y_{(i-1)l+2}) \oplus \ldots \oplus Y_{il}$
4: **end for**
5: **return** $\Lambda_\pi^{(l)}, \Lambda_\pi^{(2l)}, \ldots, \Lambda_\pi^{(Rl)}$

Figure 5 represents the process of accumulating entropy using a permutation that shifts 8-bit data to the left by 1 bit.

XOR entropy accumulation[?]

In 2023, the XOR-based entropy accumulation mechanism was proposed by Choi et al[41]. The following mechanism accumulates entropy using only XOR operations without hash functions.

Assume that the n-bit random variables $Y_1, Y_2, \ldots, Y_l$ from the noise source are independent. Denote the bit vector representation of each Y_i by $Y_i = (Y_{i1}, Y_{i2}, \ldots, Y_{in})$. Also, define $\Gamma^{(l)} = \sum_{j=1}^{l} Y_j$. For $n = 1$, by Theorem 3.2, the probability that $\Gamma^{(l)} = Y_1 \oplus Y_2 \oplus \cdots \oplus Y_l = 0$ converges to $1/2$ as l increases[42]. However, if $Y_1, Y_2, \ldots, Y_l$ are extended to $n \geq 2$ bits, while the probability of each bit being zero converges to $1/2$, the min-entropy of $\Gamma^{(l)}$ does not converge to n since the independence between the bits Y_{ij} of each random variable Y_i cannot be

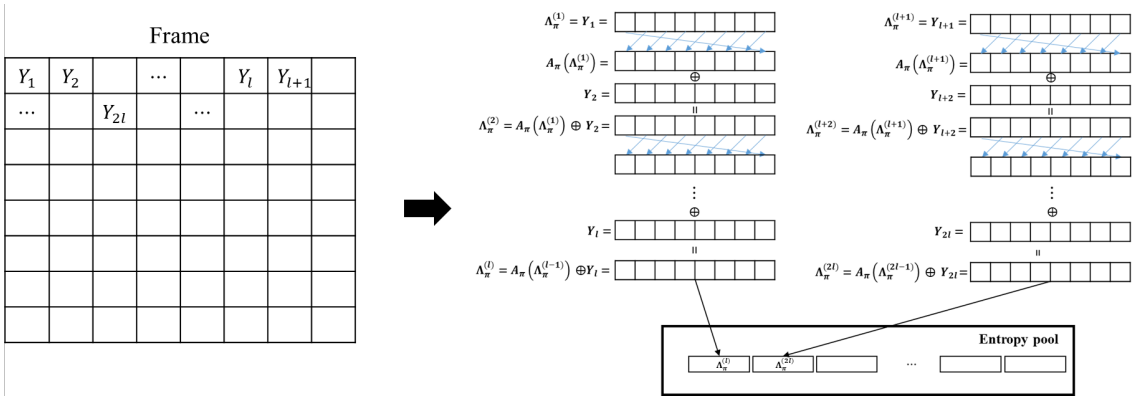


Figure 5. WRNG entropy accumulation.

guaranteed. One of the primary results from Choi *et al.*[41] is that if the distribution of n-bit random variables $Y_1, Y_2, \ldots, Y_l$ satisfies specific conditions, the probability distribution $D_{\Gamma^{(l)}}$ converges to a uniform distribution as l increases. Based on this result, Theorem 3.3 guarantees the lower bound on the min-entropy of $D_{\Gamma^{(l)}}$.

Theorem 3.2 ***(Piling-up Lemma)*** *Let Y_i $(1 \le i \le n)$ be 1-bit random variables that are independent of each other, with the probability of being 0 as p_i and the probability of being 1 as $1 - p_i$. Then, the following holds.*

$$P[Y_1 \oplus Y_2 \oplus \cdots \oplus Y_n = 0] = \frac{1}{2} + 2^{n-1} \prod_{i=1}^{n} \left(p_i - \frac{1}{2} \right).$$

Theorem 3.3 *Let $Y_1, Y_2, \ldots, Y_l$ be independent n-bit random variables, and define $\Gamma^{(l)} := Y_1 \oplus Y_2 \oplus \ldots \oplus Y_l$ and $\omega := \min\{\|D_{Y_1}\|_{\min}, \|D_{Y_2}\|_{\min}, \ldots, \|D_{Y_l}\|_{\min}\}$. Then for $\omega > 0$, the following holds:*

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

Algorithm 2 is based on the XOR entropy accumulation mechanism and uses n-bit noise sources $Y_1, Y_2, \ldots$ that satisfy the conditions of Theorem 3.3 to accumulate entropy. The algorithm in Figure 6 guarantees that all R accumulated n-bit data have an entropy of at least h per bit.

Algorithm 2 XOR entropy accumulation[41]

Input: n-bit input data $Y_1, Y_2, \cdots$, output entropy per bit h, number of n-bit output data R, the minimum value ω of the smallest probability among the distributions $D_{Y_1}, D_{Y_2}, \ldots, D_{Y_l}$
Output: accumulated entropy $\Gamma^{(l)}, \Gamma^{(2l)}, \ldots, \Gamma^{(Rl)}$

1: Calculate $l = \left\lceil \frac{1}{\ln(1 - 2^n\omega)} \cdot \ln \left(\frac{n(1-h) \cdot \ln 2}{2^n - 1} \right) \cdot \right\rceil$
2: **for** $k = 1$ to R **do**
3: Calculate $\Gamma^{(kl)} = Y_{(k-1)l+1} \oplus Y_{(k-1)l+2} \oplus \ldots \oplus Y_{kl}$.
4: **end for**
5: **return** $\Gamma^{(l)}, \Gamma^{(2l)}, \ldots, \Gamma^{(Rl)}$

Basically, both algorithms require the independence of n-bit noise sources $Y_1, Y_2, \ldots$ and additional conditions. The noise sources used in WRNG entropy accumulation should be two-monotone distributions. Additionally, the corresponding covering number for the permutation used must be finite. On the other hand, the noise sources used in the XOR entropy accumulation must have a minimum probability greater than zero. If an

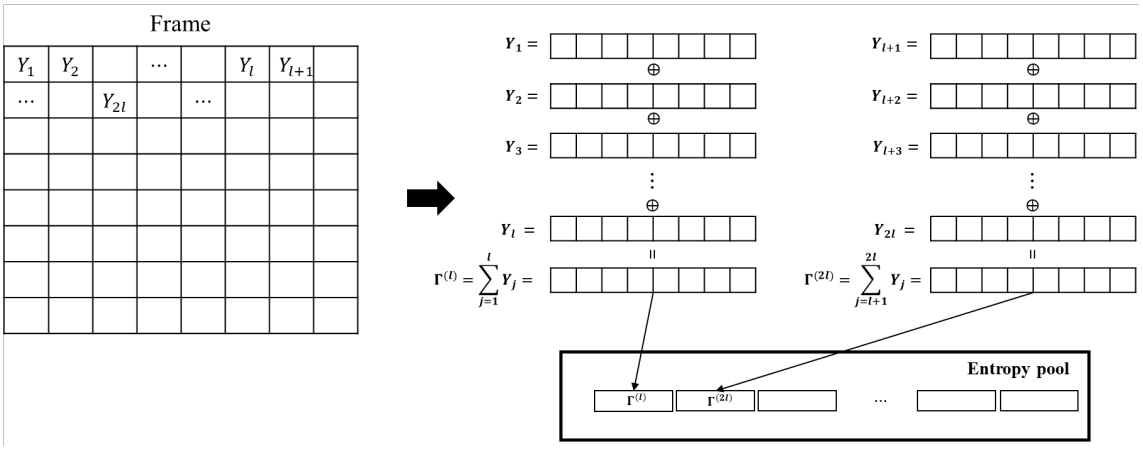
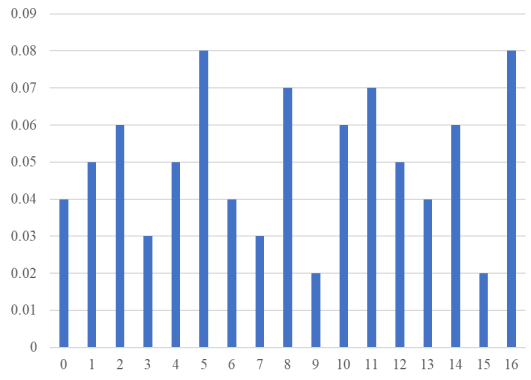
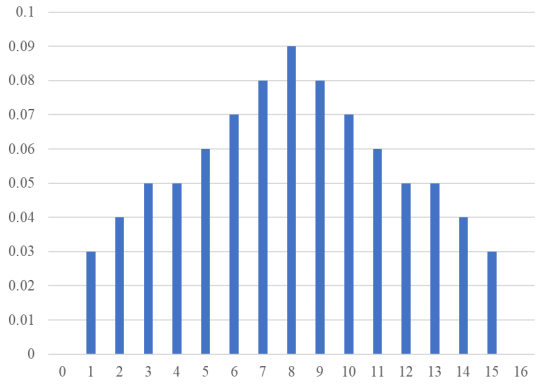


Figure 6. XOR entropy accumulation.



(a) Distribution with minimum probability ≥ 0.02.



(b) 2-monotone distribution suitable for WRNG entropy accumulation

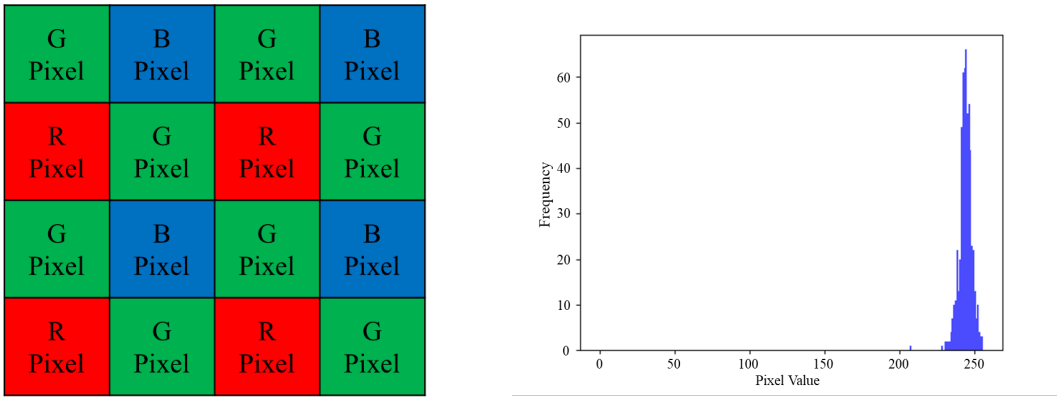
Figure 7. Pixel entropy distribution: (a) and (b) are examples of distributions suitable for applying XOR entropy accumulation and WRNG entropy accumulation, respectively.

element with a probability of zero exists in some distribution, it has to be eliminated from the sample space to apply the XOR entropy accumulation mechanism. However, if the intersection of the supports of all distributions is an empty set, the mechanism cannot be employed. Therefore, depending on the nature of the distribution, one of both entropy accumulation mechanisms can be chosen.

Figure 7 represents the distributions suitable for each mechanism. In distribution 7a, the minimum probability is greater than or equal to 0.02 allowing the application of the XOR entropy accumulation mechanism, but it is not a 2-monotone distribution, so the Window entropy accumulation mechanism cannot be applied. Additionally, the distribution 7b follows a 2-monotone distribution, allowing the application of the WRNG entropy accumulation. However, XOR entropy accumulation cannot be employed since its minimum probability is zero.

3.3. Independence of the pixels in an image sensor

Since each OBP of the image sensor does not interfere with the other during signal processing, they can be assumed to be independent[32]. Therefore, in IP cameras, the dark shot noise of the image sensor is suitable for the aforementioned entropy accumulation algorithms.



(a) RGB pixel array structure. **(b)** Distribution of Y data

Figure 8. Pixel array structure and Y data distribution: (a) shows the RGB pixel array structure in the image sensor and (b) shows the distribution of Y data for an OBP used for the independence test.

Some camera modules may not allow direct extraction of dark shot noise and only permit extraction of luminance information(called Y data) from the image. Nonetheless, that can still be used as a noise source. As shown in Figure 8a, each pixel of the image sensor is covered with a Bayer color filter, which allows only one color component to be generated per pixel. The Y data is output according to:

$$Y = \alpha * R + \beta * G + \gamma * B, \qquad (\alpha, \beta, \gamma \text{ are positive integers})$$

where R, G, and B represent 10-bit values of the surrounding red, green, and blue pixels, respectively. The Y data is converted into 8-bit data through post-processing correction. Unlike dark shot noise, the Y data is determined by the linear combination of adjacent pixel data, meaning that the independence of each OBP cannot be guaranteed. Furthermore, the post-processing correction method for image sensors depends on the manufacturer and the exact equation for Y data can be confidential. Therefore, extracting raw data of shot noise can be difficult. However, selecting non-adjacent OBPs for Y data extraction is expected to reduce the dependency between the data from spaced pixels. Therefore, to verify the independence of Y data concerning the intervals between each OBP, it is necessary to perform independence tests by gradually increasing the distance between OBPs. This process aims to determine the minimum interval that ensures independence between each data.

We demonstrate the feasibility of using Y data as a noise source through an example. Our setting for the experiment is as follows: The noise source used in the experiment consists of 15,360 pixels from the top eight rows. However, for the further experiment, only pixels that met the conditions were used instead of all the pixels. We estimated the probability distribution based on 1,500 frames of 8-bit data transmitted by each of the 15,360 pixels in the image sensor. Then, we selected only 7,513 pixels that satisfy conditions 1 and 2 of Theorem 3.1. Given 7,513 OBPs from the image sensor 'PV4209K', a complementary metal oxide semiconductor (CMOS) sensor with megapixels from Pixelplus, we performed independence tests to verify the suitability of the Y data as a noise source. Figure 8b represents a distribution of output from a randomly selected pixel among the OBPs. Most distributions from OBPs follow a similar distribution. The independence check refers to the statistical tests in 'Noise Source Entropy Verification Algorithm in Software Environment' by the Telecommunications Technology Association (TTA) standard in South Korea, which corresponds to NIST SP 800-90B discussed in the previous section. To identify mutually independent OBPs within the total OBPs of the image sensor, we selected OBPs by increasing the stride between them by one and conducted independence tests. The independence tests, at a specified significance level, attempt to discover dependencies in the data. It comprises three

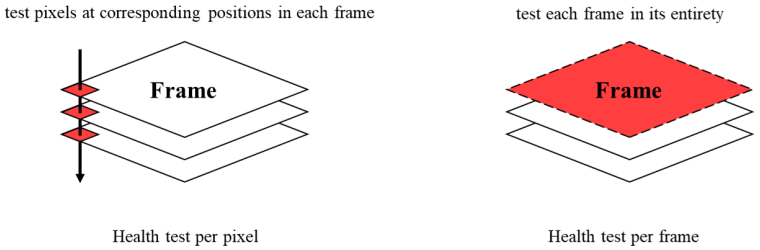


Figure 9. Health test for parallel noise sources.

stages: independence testing between two different byte sequences, between three different byte sequences, and among four byte sequences. Only if dependency cannot be identified at all three stages, can it be determined that mutual independence exists between the OBPs. We set the significance level at 0.001 to assess independence among OBPs. With a stride of 1, dependency was observed, whereas mutual independence was confirmed starting from a stride of 2. Among the 7,513 pixels used in the experiment, there are 3,782 pixels with a stride of 2 or more, and the experimental results confirm that these 3,782 pixels are mutually independent.

3.4. Health tests for parallel noise sources [43]

A noise source may produce its output stuck to a single value, or generate values with a repeated pattern due to failure or aging. To prevent such malfunctions, it is necessary to evaluate the functionality of the noise source on the fly. NIST recommends using health tests to detect noise source malfunctions and provides a set of tests for a single noise source in NIST SP 800-90B [38]. However, for the noise sources used in IP cameras, which simultaneously generate randomness using multiple OBPs within the image sensor, it is inefficient to directly apply health tests designed for a single noise source. Therefore, we introduce health tests applicable to individual noise sources and then present health tests designed for image sensors, which act as parallel noise sources. If a noise source fails at least one of these tests, it is deemed an inappropriate noise source, and the entropy harvest process is halted. Figure 9 presents a comparison between the test performed on a per-pixel basis and the test performed on a per-frame basis.

Health Tests for each OBP

Health tests for each OBP are performed at the pixel level and consist of two types of tests:

i. Repetition Count Test: This test is designed to assess whether a specific bit value appears continuously at a particular OBP position. If the same bit value appears consecutively beyond a predetermined threshold, the OBP is excluded from the noise source.
ii. Adaptive Proportion Test: This test is designed to evaluate the repetition frequency of the initial input bit value at a specific OBP position. If the bit value occurs more than a predetermined threshold, the OBP is excluded from the noise source.

These tests are designed based on NIST SP 800-90B, and any OBP that returns a failure in the test is excluded from the noise source. If the ratio of error pixels to the total number of OBPs exceeds the threshold, the image sensor is considered unsuitable as a noise source and the entropy harvesting process is inhibited.

Health Tests for entire OBP

The health test for the entire OBP is conducted on a frame-by-frame basis upon user request. Whereas the previous tests are health tests for each OBP, this health test considers the data from entire OBPs as a single frame and performs the health test on this aggregated data. This test consists of two types of evaluations, each

specifically designed for parallel noise sources.

i. Monobit Test per Frame: This test is designed to examine whether all bits in the output data from the noise source are uniformly distributed. If the distribution of bits from the noise source is skewed towards a particular value beyond a predetermined threshold, the noise source for the entire OBP is deemed inappropriate.
ii. Poker Test per Frame: This test checks whether all values output from entire OBP within a frame are uniformly distributed across intervals classified by distribution characteristics. If the output data is heavily concentrated in a specific interval, the noise source for the entire OBP is considered inappropriate.

If an OBP fails either of these evaluations consecutively beyond a predetermined threshold, the noise source is deemed unsuitable for the random number generation.

When a health test designed for a single noise source is directly applied to parallel noise sources, the examination proceeds to the next OBP only after completing the check for one OBP. Consequently, the test must continue until a certain number of malfunctioning OBPs are discovered. If the malfunctioning OBPs are clustered towards the end of the testing sequence, the detection of the noise source malfunction is delayed until reaching that position. In contrast, a health test designed using the characteristics of parallel noise sources can determine the malfunction of the entire noise source at once without checking each OBP individually, allowing for a quicker halt in the random number generation. Thus, applying these health tests on an IP camera image sensor can detect noise source malfunctions more rapidly than applying a single noise source health test while ensuring a similar assurance level of noise source. The two types of health tests were executed on a PC environment with an Intel(R) Core(TM) Ultra 7 155H CPU. For an input size of 20MB, both tests recorded an execution time of 0.005 seconds, which is sufficient to execute health tests on-the-fly even in an embedded environment.

3.5. Managing the entropy pool

The two entropy accumulation algorithms introduced earlier collect n-bit data in parallel and accumulate $R \times n$-bit data at once. Consider an array of entropy pools with D entries of n-bit pools ($D \geq R$). $D \times n$ bits are required to fill the entropy pool.

In the initial step, the first R entries are filled. The remaining entries are then filled in a round-robin fashion. When the D entropy pools are filled, they are updated with new entropy, starting with the first pool. Given that all outputs of the accumulation algorithm already satisfy a min-entropy of h per bit, close to full entropy, no additional functions for enhancing the entropy nor mixing with other entries are required. Consequently, each entropy pool operates in parallel and independently, which is referred to as a parallel entropy pool.

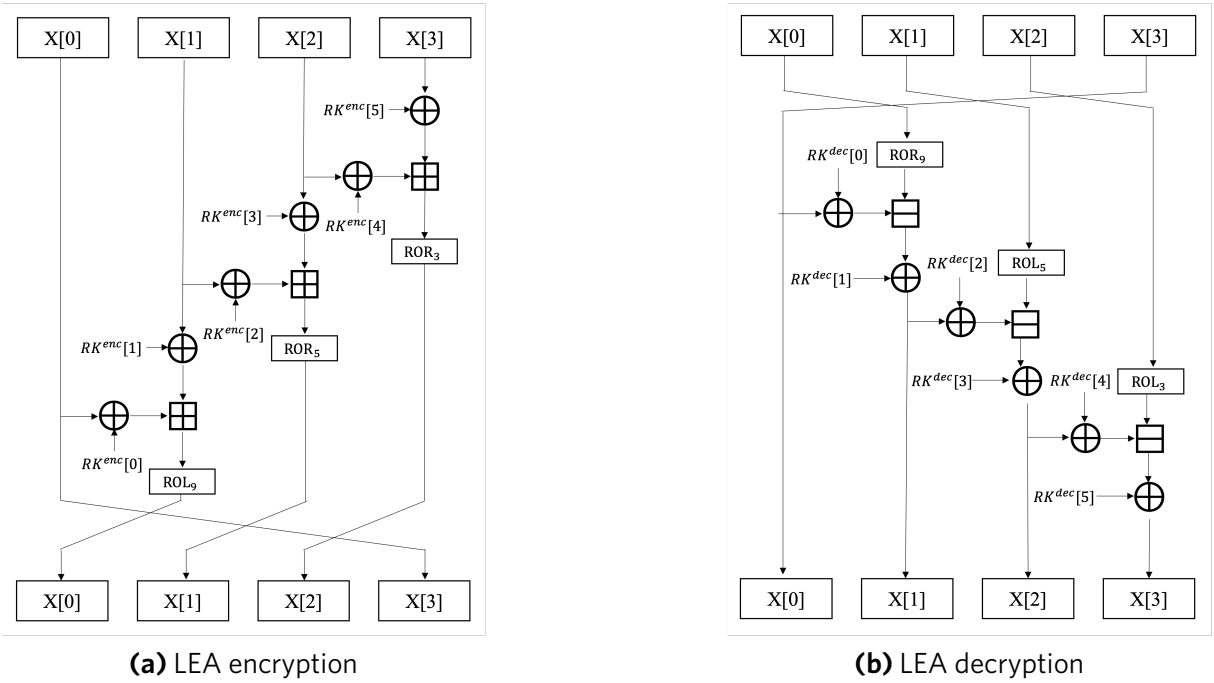
When a user requests random numbers, the required number of bits is produced from the entropy pools in a round-robin fashion over the D different pools, starting with the first pool. However, the priority of using random numbers from specific entropy pools may be determined by the entropy pool administrator. Additionally, to manage which of the D entropy pools contains sufficient entropy, the administrator can use a D-bit array to track the status of each pool.

4. KEY DERIVATION AND EXPERIMENTAL RESULTS

In Section "ENTROPY HARVEST", we discuss how to guarantee the lower bound of accumulated entropy in the pool and propose an efficient method with simple bitwise operations. In order to generate encryption keys or sensitive security parameters, we need a cryptographically secure PRNG that takes input seeds from the entropy pool and produces pseudorandom bits on request. Additionally, depending on the environment of applications, a key derivation function (KDF) can be optionally employed to enhance security. This section explains the general structure of PRNGs and KDFs suitable for IP cameras and highlights the importance of

Table 2. LEA specification

	Block length(bytes)	Key length(bytes)	Number of rounds
LEA-128	16	16	24
LEA-192	16	24	28
LEA-256	16	32	32



(a) LEA encryption **(b)** LEA decryption

Figure 10. Structure of one round of the block cipher LEA: (a) represents the one round encryption and (b) represents the one round decryption. These algorithms are inverses of each other[44].

the secure random bit generation free from backdoor concerns. We propose a set of lightweight schemes for encrypting video data and key derivation in IP cameras. We also discuss the implementation of our encryption scheme in a commercial IP camera.

4.1. Cipher suites for IP cameras

Lightweight block Cipher LEA

The block cipher Lightweight Encryption Algorithm (LEA) is an efficient encryption algorithm in constrained environments. LEA-128 is about 1.7 times faster than AES-128 on ARM platform[44]. It was established as a national standard of South Korea (KS X 3246) in 2013 and as an ISO/IEC standard for lightweight block ciphers (ISO/IEC 29192-2:2019) in 2019. LEA is designed with the ARX (Addition, Rotation, XOR) operation to manage resources efficiently in a software environment. The modular addition is the only non-linear part, replacing the S-box of AES. The ARX structure allows the use of basic instructions provided by the processor, resulting in fast computation speeds, and does not require additional memory to store non-linear substitution tables. Depending on the key size for the required security level, the specifications of LEA are categorized as LEA-128, LEA-192, or LEA-256, as shown in Table 2[44].

Figure 10 illustrates the structure of the LEA round function used in the encryption and decryption processes. The input to the round function is utilized in the internal operations as an array of 32-bit words. The internal operations constituting the round function are defined as follows: $\boxplus$ denotes 32-bit addition, $\oplus$ represents bitwise exclusive OR (XOR), and ROL_n (ROR_n) indicates n-bit left (right) rotation operations, respectively.

PRNG

A PRNG is a deterministic algorithm to generate pseudorandom numbers[45]. The input seed of a PRNG should contain sufficient entropy to generate secure random bits. A PRNG takes an input seed from the entropy pool which is preferably close to full entropy. To condense data from the pool to a seed, we can apply one of the six vetted conditioning functions specified in NIST SP 800-90C to ensure the entropy level in the seed[46]. Considering the cryptographic module and its operating environment, we choose one of three PRNG mechanisms specified in NIST SP 800-90A.

Considering the efficient implementation of the cryptographic module using block ciphers, we choose the Counter Mode Deterministic Random Bit Generator (CTR-DRBG) with LEA as its primitive. It has a small memory footprint, making it well-suited for lightweight environments such as IP cameras. Additionally, it can achieve speed enhancements through parallel implementation.

KDF

The KDF is the final step in creating an encryption key. When mapping such data into bitstrings of the required length through truncation or other heuristic methods, entropy loss may occur, potentially amplifying bias. Therefore, there is a need for a KDF to generate keys of desired length with high entropy[47]. The KDF mechanisms are specified as one-step key derivation, or two-step key derivation by NIST[48].

4.2. Secure implementation of PRNGs

In a black-box model, a PRNG may contain a Trojan horse which performs a SETUP attack to obtain the secret key information via subliminal channels. We emphasize the importance of secure PRNG implementations by showing a SETUP attack example proposed by Choi et al[49].

Algorithm 3 CRYSTALS-Kyber : key generation

Input: k, η, d_t

Output: $pk := (\vec{t}, \rho), sk := \vec{s}$

1: $\rho, \sigma \xleftarrow{\$} \{0,1\}^{256}$

2:

3: $\vec{A} \sim R_q^{k \times k} := Sam(\rho)$

4:

5: $(\vec{s}, \vec{e}) \sim \beta_\eta^k \times \beta_\eta^k := Sam(\sigma)$

6:

7: $\vec{t} := Compress_q(\vec{A}\vec{s} + \vec{e}, d_t)$

8:

9: **return** $pk := (\vec{t}, \rho), sk := \vec{s}$

CRYSTALS-Kyber is a post-quantum cryptographic scheme for key encapsulation mechanisms selected as a NIST standard[50]. Algorithm 3 below gives the early version[51] of its key generation algorithm vulnerable to SETUP attacks, where the random numbers are provided by the Trojan horse in PRNG without modifying CRYSTALS-Kyber algorithm. The core of this attack lies in modifying the random number generator such that it does not independently generate ρ and σ, but rather generates ρ first and then encrypts ρ with the attacker's public key to produce σ. Choi *et al.* present an attack to extract the private key from the corresponding public key satisfying the conditions of SETUP attack[49]. The latest version of CRYSTALS-Kyber has been updated to generate ρ and σ from a single random number in a specified manner. Therefore, when implementing an RNG, we have to fully investigate its internal algorithm.

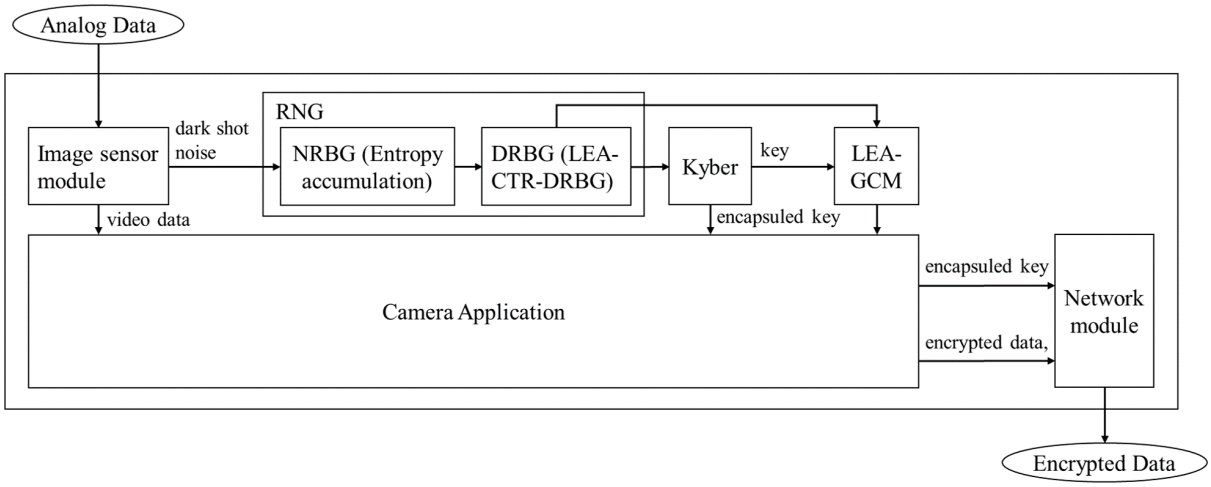


Figure 11. Structure of secure IP camera.

4.3. Performance evaluation of IP camera

We propose an encryption scheme for video data in IP cameras depicted in Figure 11. We experimentally verify the performance using the image sensor *PV4209K* integrated into the camera hardware module with *Cortex-A53* processor. To implement an IP camera free from subliminal channels, we disclose the structure of each component and verify its operation to avoid the use of black-box modules.

Entropy harvest from image sensors

The FHD image sensor *PV4209K* utilizes the top eight rows, comprising 1920×8 OBPs, as noise sources, while encrypting the image of 1920×1080 pixels excluding the OBPs. The image sensor of the IP camera that we use in the experiment cannot directly extract dark shot noise but outputs only the luminance information Y data of the image. The Y data is a linear combination of information from adjacent pixels as described in Section "ENTROPY HARVEST". In the environments, the process of generating random numbers and deriving keys proceeds as follows: First, we estimate the probability distribution of 15,360 OBPs from the top eight rows of the image sensor used as noise sources. To estimate the OBP distribution, we utilized 8-bit data from 15,000 frames, satisfying two specific properties from 7,513 OBPs.

1. The distribution follows a 2-monotone distribution.
2. The distribution has a min-entropy of at least 2.

Y data depends on adjacent OBPs, so independence testing is necessary. Based on the independence test in Section "ENTROPY HARVEST", we select 3,782 OBPs with stride of 2 among the 7,513 OBPs.

The distribution of Y data obtained from discriminated OBPs satisfies all three assumptions of WRNG entropy accumulation mechanism but does not satisfy those of XOR entropy accumulation mechanism. Therefore, we employ Algorithm 1 as the entropy accumulation mechanism. Higher min-entropy of the noise source distribution contributes to the efficiency of entropy accumulation. In this experiment, we statistically check the min-entropy of each OBP to be at least 2. With $k' = \left\lfloor \frac{k}{2} \right\rfloor = 1$, we use the permutation $\pi = rot_{(1,8)}$ where the covering number $m = C_{\pi,k'} = 8$.

We aim to achieve entropy per bit of 0.99 or higher, and Algorithm 1 performs 62 XOR operations for accu-

Table 3. Entropy accumulation speed

Test	1	2	3	4	5	6	7	Average
Bntropy accumulation speed(bps)	7,321	7,321	7,321	7,320	7,325	7,320	7,332	7,323

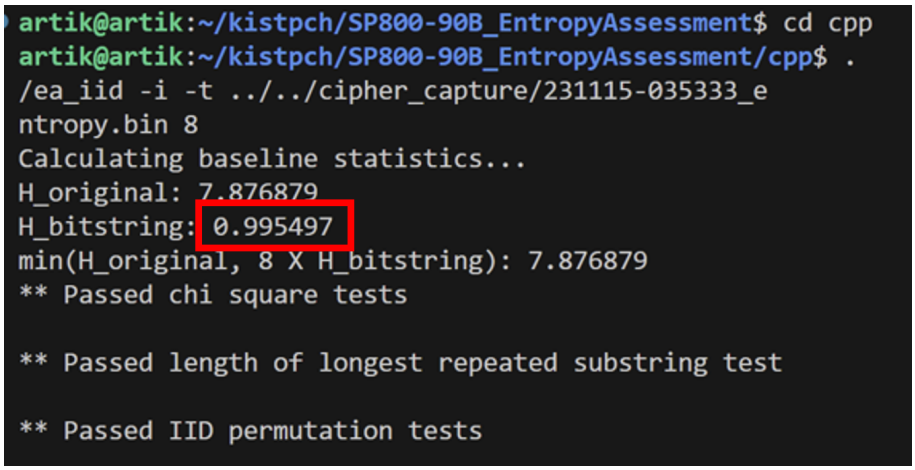


Figure 12. Min-entropy test result.

mulation with

$$l = \left\lceil m \left[1 - \frac{2}{k} \left(\frac{\ln (1-h)}{\ln 2} \right) \right] \right\rceil = 62.$$

Table 3 presents the experimental results of entropy accumulation speed. Iterating seven times, we confirm that random numbers are generated at an average speed of 7.3 kbps.

The camera used in the experiment outputs 15 frames per second, allowing for the generation of 915 8-bit random numbers that satisfy a min-entropy of 0.99. When these outputs are used as the seeds for the PRNG, considering the operation speed of the RNG, it is sufficient to provide random bits without delay for updating symmetric keys every minute. Furthermore, using the entropy estimation algorithm from NIST SP 800-90B, we confirmed that the min-entropy per bit in the entropy pool, generated using the accumulation mechanism, is greater than 0.99, as shown in Figure 12 [38].

On the other hand, it was demonstrated that when utilizing a noise source that satisfies the conditions of Theorem 3.3, the output from the XOR entropy accumulation algorithm has a min-entropy exceeding 7.86 per byte [41].

Video encryption

We have selected LEA-CTR-GCM as encryption algorithm for the video data. Galois/Counter Mode (GCM) is a block cipher mode of operation that provides authenticated encryption using a binary Galois field for universal hashing. GCM mode ensures confidentiality, integrity, and authentication altogether. Using CTR mode enables parallel implementation even in a constrained environment such as IP cameras.

Figure 13 illustrates the demonstration of video encryption in an IP camera. It shows both the encrypted video received from the camera and the decrypted video.

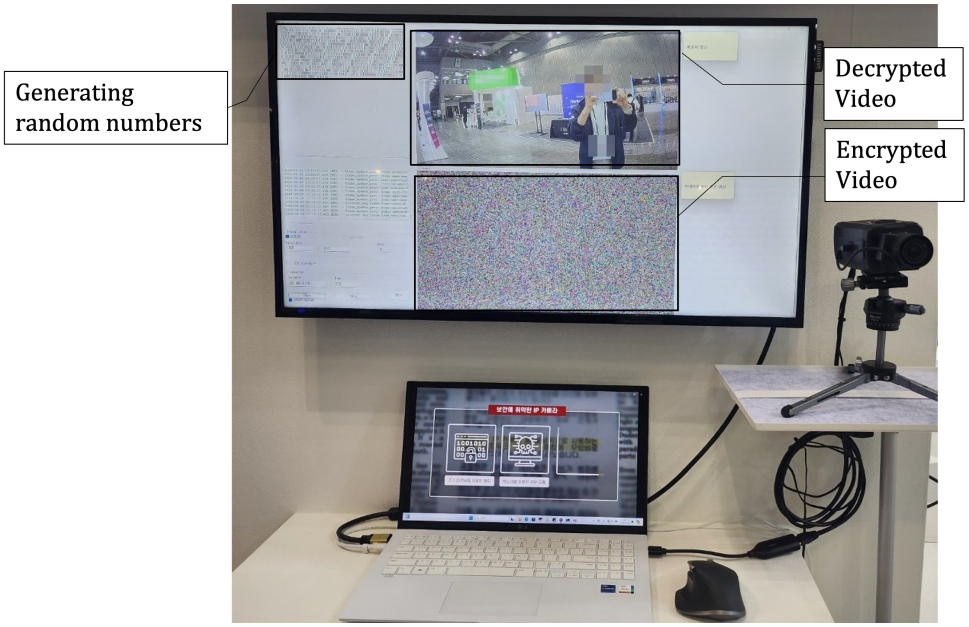


Figure 13. Demonstration of video encryption.

5. CONCLUSION

In video surveillance systems, we expect the IP cameras to be free from hacking or vulnerabilities. Even though suspicious IP cameras are still in use, it is difficult to detect and remove backdoors residing in the black-box component. In this paper, we propose that every module of an IP camera must be tested and no black-box component should be used to allow scanning for vulnerable components and take verifications through security evaluation processes on behalf of the users. Therefore, we categorize the potential vulnerabilities that may exist in each module and suggest an encryption scheme suitable for IP cameras and focus on the RNG module known to be a particularly vulnerable component, where attacks such as subliminal channels may be ambushed. For a secure and efficient RNG configuration, we show that the randomness of the image sensor can be extracted in parallel as a noise source without any additional hardware RNG. The min-entropy is estimated and accumulated in the entropy pool. For this purpose, we suggest two high-speed entropy accumulation mechanisms that ensure the lower bound of min-entropy using only bitwise operations.

Our encryption scheme is applied to the FHD camera module with a built-in *Cortex-A53*-based image sensor *PV4209K*. The scheme can harvest a min-entropy of 0.99 per bit from the OBPs in the image sensor at a rate of 7.32 kbps. With the lightweight block cipher LEA, an encryption scheme is constructed that uses LEA-CTR-DRBG and LEA-GCM. It was demonstrated that our scheme can encrypt video data at an average of 15 frames per minute, with key updates occurring every 60 seconds.

6. DECLARATIONS

6.1. Authors' contributions

Designing the main idea and concept: Ryu J, Kang J, Yeom Y
Drafting the manuscript: Ryu J, Kim G, Ko J
Setting up the experimental environment: Kang D
Collecting and analyzing the experiment data: Kim G, Ko J

6.2. Availability of data and materials
Not applicable.

6.3. Financial support and sponsorship
This work was supported by the Ministry of Science and ICT (MSIT), South Korea (COMPA2024SCPO_B_0210, Development of Secure IP Camera Based on Quantum Technology), the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korean government (MSIT) (No. 2710006735, Development of Core Technology to Produce KCMVP Security High-Level 3 Cryptographic Module) and the IITP grant funded by the Korean government (MSIT) (No. 2021-0-00046, Development of Next-Generation Cryptosystem to Improve Security and Usability of the National Information System).

6.4. Conflicts of interest
Dongkeun Kang is affiliated with SDT Inc., while the other authors have declared that they have no conflicts of interest.

6.5. Ethical approval and consent to participate
Not applicable.

REFERENCES
1. Mordor intelligence research & advisory. video surveillance system market size & share analysis - growth trends & forecasts (2024 - 2029). Mordor Intelligence; 2023. Available from: https://www.mordorintelligence.com/industry-reports/video-surveillance-systems-market. [Last accessed on 25 Dec 2024]
2. Tech Report. 15+ Surveillance Camera Statistics (2023 Updated Data); 2023. Accessed: 2024-06-10. Available from: https://techreport.com/statistics/hardware-gadgets/surveillance-camera-statistics/. [Last accessed on 25 Dec 2024]
3. Market us. IP camera market; 2023. Accessed: 2024-06-10. Available from: https://market.us/report/ip-camera-market/. [Last accessed on 25 Dec 2024]
4. BrandEssence market research. IP camera market size, share & trends analysis Report; 2022. Accessed: 2024-06-10. Available from: https://brandessenceresearch.com/technology-and-media/ip-camera-market-latest. [Last accessed on 25 Dec 2024]
5. Verified market research. Global IP camera market size; 2024. Accessed: 2024-06-10. Available from: https://www.verifiedmarketresearch.com/product/ip-camera-market/. [Last accessed on 25 Dec 2024]
6. Global market insights. IP camera market; 2023. Accessed: 2024-06-10. Available from: Available from: https://www.gminsights.com/industry-analysis/ip-camera-market. [Last accessed on 25 Dec 2024]
7. MSIT. The 13th Meeting of the ICT Strategy Committee Was Held; 2019.
8. CISA. Dahua Technology Co., Ltd digital video recorders and IP cameras; 2017. Accessed: 2024-06-10. Available from: https://www.cisa.gov/news-events/ics-advisories/icsa-17-124-02. [Last accessed on 25 Dec 2024]
9. Kovacs E. Backdoor found in dahua video recorders, cameras; 2017. Accessed: 2024-06-10. Available from: https://www.securityweek.com/backdoor-found-dahua-video-recorders-cameras. [Last accessed on 25 Dec 2024]
10. Brook C. Dahua patching backdoor in DVRs, IP cameras. Threatpost; 2017. Accessed: 2024-06-10. Available from: https://threatpost.com/dahua-patching-backdoor-in-dvrs-ip-cameras. [Last accessed on 25 Dec 2024]
11. Domonoske C. S.C. Mom says baby monitor was hacked; experts say many devices are vulnerable. NPR; 2018. Accessed: 2024-06-10. Available from: https://www.npr.org/sections/thetwo-way/2018/06/05/617196788/s-c-mom-says-baby-monitor-was-hacked-experts-say-many-devices-are-vulnerable. [Last accessed on 25 Dec 2024]
12. Chokshi N. Somebody's watching: hackers breach ring home security cameras; 2019. Accessed: 2024-06-10. Available from: https://www.nytimes.com/2019/12/15/us/Hacked-ring-home-security-cameras.html. [Last accessed on 25 Dec 2024]
13. PW consulting. Video surveillance market; 2022. Accessed: 2024-06-10. Available from: https://pmarketresearch.com/video-surveillance-market/. [Last accessed on 25 Dec 2024]
14. Grand view research. Video surveillance market size, share & trends analysis report; 2023. Accessed: 2024-06-10. Available from: https://www.grandviewresearch.com/industry-analysis/video-surveillance-market-report. [Last accessed on 25 Dec 2024]
15. 6Wresearch. China video surveillance systems market (2019-2025); 2023. Accessed: 2024-06-10. Available from: https://www.6wresearch.com/industry-report/china-video-surveillance-systems-market-2019-2025. [Last accessed on 25 Dec 2024]
16. Doffman Z. Warning as eavesdropping risk hits millions Of Chinese-Made cameras; 2019. Accessed: 2024-06-10. Available from: https://www.forbes.com/sites/zakdoffman/2019/08/03/update-now-warning-as-eavesdropping-risk-hits-millions-of-chinese-mad

e-cameras/?sh=18de3d66bf2c. [Last accessed on 25 Dec 2024]
17. Fingas J. Army base pulls Chinese security cameras over ”negative perception”. Engadget; 2018. Accessed: 2024-06-10. Available from: https://www.engadget.com/2018-01-15-army-base-pulls-chinese-security-cameras.html. [Last accessed on 25 Dec 2024]
18. Federal Communications commission. FCC bans authorizations for devices that pose national security threat; 2022. Accessed: 2024-06-10. Available from: https://www.fcc.gov/document/fcc-bans-authorizations-devices-pose-national-security-threat. [Last accessed on 25 Dec 2024]
19. Martin N. US bans Chinese telecom, surveillance cameras; 2022. Accessed: 2024-06-10. Available from: https://www.dw.com/en/us-bans-chinese-telecom-surveillance-cameras/a-63895206. [Last accessed on 25 Dec 2024]
20. Morrison R. UK government ban for Chinese Hikvision CCTV cameras. Tech Monitor; 2022. Accessed: 2024-06-30. Available from: https://techmonitor.ai/government-computing/hikvision-ban-uk-government-oliver-dowden. [Last accessed on 25 Dec 2024]
21. Blake E. Government department removes Chinese cameras that caught Matt Hancock’s affair; 2022. Accessed: 2024-06-10. Available from: https://www.standard.co.uk/news/uk/hikvision-chinese-cameras-cctv-dwp-banned-matt-hancock-affair-b1008586.html. [Last accessed on 25 Dec 2024]
22. National intelligence service, Republic of Korea. Security Conformance Verification Policy for Product Family of Image Information Processing Devices in Public Sector; 2024. Accessed: 2024-06-10. Available from: https://www.ncsc.go.kr:4018/main/cop/bbs/selectBoardArticle.do?bbsId=SecurityNotification_main&nttId=127788&menuNo=030000&subMenuNo=031200&thirdMenuNo=. [Last accessed on 25 Dec 2024]
23. Simmons GJ. The history of subliminal channels. *IEEE J Sel Areas Commun* 1998;16:452–62. DOI
24. Lampson BW. A note on the confinement problem. *Commun ACM* 1973 oct;16:613–15. DOI
25. Petitcolas FA, Anderson RJ, Kuhn MG. Information hiding-a survey. *Proc IEEE* 1999;87:1062–78. DOI
26. Cheddad A, Condell J, Curran K, Mc Kevitt P. Digital image steganography: survey and analysis of current methods. *Signal Process* 2010;90:727–52. DOI
27. Girling CG. Covert channels in LAN’s. *IEEE Trans Softw Eng* 1987;13:292. DOI
28. Simmons GJ. The prisoners’ problem and the subliminal channel. In: Advances in Cryptology: Proceedings of Crypto 83. Springer; 1984. pp. 51–67. DOI
29. Young A, Yung M. Malicious cryptography: exposing cryptovirology. John Wiley & Sons; 2004. DOI
30. National Institute of Standards and Technology. A statistical test suite for random and pseudorandom number generators for cryptographic applications. National Institute of Standards and Technology; 2010. SP 800-22 Rev. 1a. Accessed: 2024-06-10. DOI
31. Telecommunications Technology Association. Guideline for the collection and application of noise source on operating systems. Telecommunications Technology Association; 2020. TTAS.KO-12.0235/R2. Available from: https://committee.tta.or.kr/data/standard_view.jsp?commit_code=PG504&pk_num=TTAK.KO-12.0235%2FR2. [Last accessed on 25 Dec 2024]
32. Park BK, Park H, Kim YS, et al. Practical true random number generator using CMOS image sensor dark noise. *IEEE Access* 2019;7:91407–13. DOI
33. Sanguinetti B, Martin A, Zbinden H, Gisin N. Quantum Random Number Generation on a Mobile Phone. *Phys Rev X* 2014;4:031056. DOI
34. Kelsey J, Schneier B, Ferguson N. Yarrow-160: notes on the design and analysis of the yarrow cryptographic pseudorandom number generator. In: International Workshop on Selected Areas in Cryptography. Springer; 1999. pp. 13–33. DOI
35. Alzhrani K, Aljaedi A. Windows and linux random number generation process: A comparative analysis. *Int J Comput Appl* 2015;113. DOI
36. Ferguson N. The windows 10 random number generation infrastructure. *Microsoft Corporation: Redmond, WA, USA* 2019. Available from: https://download.microsoft.com/download/1/c/9/1c9813b8-089c-4fef-b2ad-ad80e79403ba/Whitepaper%20-%20The%20Windows%2010%20random%20number%20generation%20infrastructure.pdf. [Last accessed on 25 Dec 2024]
37. Dodis Y, Guo S, Stephens-Davidowitz N, Xie Z. No time to hash: On super-efficient entropy accumulation. In: Advances in Cryptology–CRYPTO 2021: 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16–20, 2021, Proceedings, Part IV 41. Springer; 2021. pp. 548–76. DOI
38. Turan MS, Barker E, Kelsey J, et al. Recommendation for the Entropy Sources Used for Random Bit Generation. Special Publication (NIST SP), National Institute of Standards and Technology, Gaithersburg, MD; 2018. DOI
39. ISO/IEC JTC 1 W SC 27. Information technology - security techniques - test and analysis methods for random bit generators within ISO/IEC 19790 and ISO/IEC 15408. ISO/IEC 20543:2019; 2019. Available from: https://www.iso.org/standard/68296.html. [Last accessed on 25 Dec 2024]
40. Shoup V. A computational introduction to number theory and algebra. Cambridge, UK: Cambridge University Press; 2009. DOI
41. Choi Y, Yeom Y, Kang JS. Practical entropy accumulation for random number generators with image sensor-based quantum noise sources. *Entropy* 2023;25. DOI
42. Matsui M. Linear cryptanalysis method for DES cipher. In: Helleseth T, editor. Advances in Cryptology — EUROCRYPT ’93. Berlin, Heidelberg: Springer Berlin Heidelberg; 1994. pp. 386–97. DOI
43. Yoo H. Design and implementation of cryptographic random number generators based on parallel noise source; 2022. Available from: https://www.riss.kr/link?id=T16633914. [Last accessed on 25 Dec 2024]
44. Hong D, Lee JK, Kim DC, et al. LEA: A 128-bit block cipher for fast encryption on common processors. In: Information Security Applications: 14th International Workshop, WISA 2013, Jeju Island, Korea, August 19-21, 2013, Revised Selected Papers 14. Springer; 2014. pp. 3–27. Available from: 10.1007/978-3-319-05149-9_1.
45. Barker E, Kelsey J. Recommendation for random number generation using deterministic random bit generators. National Institute of

Standards and Technology; 2015. SP 800-90A Rev. 1. Accessed: 2024-06-10. DOI

46. Barker E, Kelsey J, McKay K, Roginsky A, Turan MS. Recommendation for Random Bit Generator (RBG) Constructions. Special Publication (NIST SP), National Institute of Standards and Technology, Gaithersburg, MD; 2022. DOI
47. Stinson D. Cryptography theory and practice. Chapman and Hall, CRC; 2005. DOI
48. Barker E, Chen L. Recommendation for key-derivation methods in key-establishment schemes. National Institute of Standards and Technology; 2011. 800-56C Rev. 1. Accessed: 2024-06-10. Available from: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Cr2.pdf. [Last accessed on 25 Dec 2024]
49. Choi Y, Yeom Y, Kang JS. Subliminal channels of CRYSTALS-Kyber applicablein the black box cryptographic module and potential security threats of IP camera; 2023. DOI
50. National Institute of Standards and Technology. Module-lattice-based key-encapsulation mechanism standard. National institute of standards and technology, gaithersburg, MD; 2023. DOI
51. Bos J, Ducas L, Kiltz E, et al. CRYSTALS - Kyber: A CCA-secure module-lattice-based KEM; 2018. pp. 353–67. DOI