information

MDPI

Article

Secure Data Transmission Using GS3 in an Armed Surveillance System

Francisco Alcaraz-Velasco, José M. Palomares, Fernando León-García and Joaquín Olivares *

Department of Electronic and Computer Engineering, Universidad de Córdoba, 14071 Córdoba, Spain; i52alvef@uco.es (F.A.-V.); jmpalomares@uco.es (J.M.P.); fernando.leon@uco.es (F.L.-G.)
* Correspondence: olivares@uco.es

Abstract

Nowadays, the evolution and growth of machine learning (ML) algorithms and the Internet of Things (IoT) are enabling new applications. Smart weapons and people detection systems are examples. Firstly, this work takes advantage of an efficient, scalable, and distributed system, named SmartFog, which identifies people with weapons by leveraging edge, fog, and cloud computing paradigms. Nevertheless, security vulnerabilities during data transmission are not addressed. Thus, this work bridges this gap by proposing a secure data transmission system integrating a lightweight security scheme named GS3. Therefore, the main novelty is the evaluation of the GS3 proposal in a real environment. In the first fog sublayer, GS3 leads to a 14% increase in execution time with respect to no secure data transmission, but AES results in a 34.5% longer execution time. GS3 achieves a 70% reduction in decipher time and a 55% reduction in cipher time compared to the AES algorithm. Furthermore, an energy consumption analysis shows that GS3 consumes 31% less power than AES. The security analysis confirms that GS3 detects tampering, replaying, forwarding, and forgery attacks. Moreover, GS3 has a key space of 2^{544} permutations, slightly larger than those of Chacha20 and Salsa20, with a faster solution than these methods. In addition, GS3 exhibits strength against differential cryptoanalysis. This mechanism is a compelling choice for energy-constrained environments and for securing event data transmissions with a short validity period. Moreover, GS3 maintains full architectural transparency with the underlying armed detection system.

Keywords: fog computing; security; Internet of Things; lightweight cipher; YOLO; armed people detection

check for updates

Academic Editors: Francesco Isgrò, Huiyu Zhou and Daniele Ravi

Received: 18 May 2025
Revised: 9 June 2025
Accepted: 13 June 2025
Published: 23 June 2025

Citation: Alcaraz-Velasco, F.; Palomares, J.M.; León-García, F.; Olivares, J. Secure Data Transmission Using GS3 in an Armed Surveillance System. *Information* **2025**, *16*, 527. https://doi.org/10.3390/info16070527

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

The detection and identification of unauthorised armed people is relevant for the security of communities, as they pose a potential threat. They have to be considered in certain public places where crowds are present (crowded streets, avenues, airports, and bus and train stations) or where there are no counterprotection measures, such as schools, universities, religious temples, etc. [1,2]. As claimed in the Global Terrorism Index study [3], the number of deaths caused by terrorism has increased by 22% to 8352, the highest level since 2017. Therefore, surveillance systems are used to address these effects. However, traditional closed-circuit television (CCTV) systems involve the supervision of large numbers of cameras by human controllers. This may result in delayed responses or even missed threats. If these systems were automated, the processing capacity data would be much greater than a person could supervise. Thus, weapon detection systems (WDS)

based on machine learning (ML) algorithms could play a crucial role in reducing armed terrorist events and enhancing citizen security.

Nevertheless, it is necessary to integrate several enabling technologies and computing models. The first is the Internet of Things (IoT) [4]. Increasingly, gadgets and devices are being connected to sense, process, send, and share data via the Internet, providing benefits to society, such as smart cities, smart grids, or smart health [5,6]. The second is a cloud computing architecture [7]. IoT devices generate a huge number of data that are transferred to the cloud for data processing, analysis, and decision making. However, this computing model causes unnecessary data transfer and cloud bottlenecks and increases the network latency. These weaknesses could be resolved with fog computing and edge computing paradigms [8]. Distributed computing models process data close to edge sensor devices. Therefore, systems that are highly scalable and have lower network latency can be developed. Typically, edge and fog devices are resource-constrained in terms of power consumption, memory, and computing capabilities. Because of this, they can be vulnerable to security gaps, which threats could exploit [9]. Therefore, developing protocols and security solutions is a challenge. We first introduced a lightweight security mechanism for data protection in [10], which evolved into the GS3 proposal presented in [11] to secure sensitive data communications.

Nowadays, the evolution and growth of artificial intelligence (AI) and machine learning (ML) algorithms are enabling new applications and transforming our lives, such as in the medical field [12], autonomous vehicles [13], or the recognition of aquatic animals [14]. The detection and recognition of armed persons is another area of application. Studies and systems have been proposed [15,16]. Nevertheless, these proposals do not address either the scalability, distributed computation, and reduced computational resources in the edge layer or security issues in communications to protect restricted data. In [17], the SmartFog mechanism is proposed to identify armed persons by efficiently utilising computational resources and communication bandwidth in the edge and fog layers.

We highlight that the main aim of this work is to evaluate the GS3 proposal in real environments. Our main motivation is to protect transmitted confidential data about possible armed persons, integrating the SmartFog [17] and GS3 [11] mechanisms. Moreover, we show that the overhead of this integration is negligible. In addition, we contrast and compare the overhead introduced by the Advanced Encryption Standard (AES) [18] with that of the GS3 method. To the best of our knowledge, this is the first work that simultaneously addresses armed people detection, distributed fog computing for low-latency decision making, and lightweight secure data transmission.

This paper is presented as follows. We first provide an introduction to establish the scope, motivations, and contributions in Section 1. Section 2 presents the reviewed literature applied to this work. The proposal is described in Section 3. Then, Section 4 shows the hardware and software platform used in this work. Our approach is evaluated by measuring the CPU, GPU, memory, execution time, and power consumption parameters in Section 5. Section 6 includes a comparative analysis of the results. Finally, Section 7 concludes this work.

2. Foundations

This section aims to provide a concise description of the reviewed security literature and the technologies applied to this proposal.

2.1. Computing Paradigms

Figure 1 shows the computing paradigms converted in this proposal.

- Edge layer: This layer is composed of billions of IoT devices, which capture and preprocess data. Moreover, it is possible to design ad hoc and dynamic wireless sensor networks (WSNs) [19]. A common characteristic in this layer is a restriction on computational resources.
- Fog layers: These intermediate layers, located between end devices and the cloud, help to resolve problems such as network latency or real-time analytics [20].
- Cloud computing: A paradigm based on powerful computing devices that provide security and enable the analysis and processing of vast IoT data. However, some applications require real-time processing and responses that this layer cannot provide.

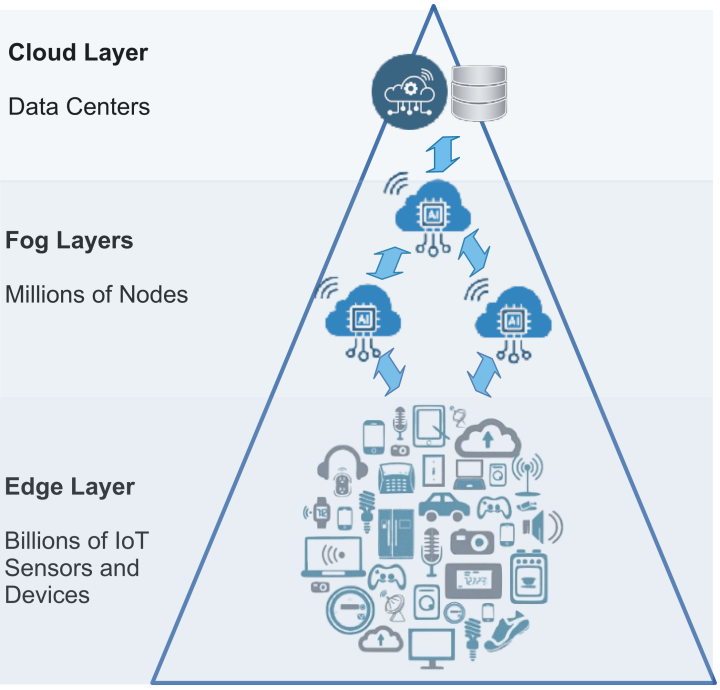


Figure 1. Computing paradigms.

2.2. Reviewed Lightweight Cipher Algorithms

This section presents a compact review of cipher mechanisms to develop the GS3 method [11]. Firstly, public key ciphers, while providing a high security level, are discarded due to high computational costs. As a consequence, asymmetric cryptography is hardly suitable for sensor and edge devices. Nevertheless, solutions such as quantum cryptography [21] try to reduce the computational costs but require highly specialised hardware. Therefore, we focus on private key ciphers, particularly lightweight private key ciphers. Symmetric ciphers can be classified as follows.

2.2.1. Stream Cipher Algorithms

Stream cipher algorithms encrypt plain data bit by bit with pseudo-random encryption sequences. The algorithms responsible for generating these sequences are named pseudo-random number generators (PRGNs). Therefore, the design and implementation of PRNGs are the focus of research in the field of cryptography. PRGNs need a private initial seed to produce an indistinguishable sequence of bits. Possible strategies to produce these sequences are the following.

1. Feedback Shift Register (FSR) Sequences: A linear FSR is a shift register whose input bit is a linear function of its previous state. The XOR function is commonly used. Linear FSRs are defined by primitive polynomials. Nonetheless, their linearity is a

security weakness. Algorithms such as Trivium [22], A5/1 [23], or Geffe [24] aim for greater security strength by connecting several linear FSRs to achieve a non-LFSR.

2. Cellular Automata: A cellular automaton (CA) is a grid of cells, each of which has a finite number of states and a neighbourhood to interact with. In each iteration, a cell calculates its new state, which depends on its state and its neighbors' states. Class 3 (chaos) patterns, as defined by Wolfram [25], evolve through chaotic behaviour. Therefore, Class 3 is interesting from a cryptography point of view. A simple one-dimensional CA would be defined with $radio = 1$ for the surrounding (adjacent) cells, so there are 2^3 possible patterns for a neighbourhood. Therefore, some of the possible 2^8 rules show chaotic behaviour. Rule-30 is a case that is defined by (1). $s_i(t)$ is the state of cell i at time t. Pentavium [26] is an improved CA with a 5-neighbourhood design to enhance the cryptographic properties of Trivium.

$$s_i(t+i) = s_{(i-1)}(t) \oplus (s_i(t) + s_{(i+1)}(t)) \quad (1)$$

3. Chaotic Systems (CS): CS are deterministic nonlinear systems that manifest complex behaviour. Moreover, tiny differences in the initial conditions provoke different behaviours. The motion of a magnetic pendulum is a simple system that exhibits chaotic behaviour. The logistic map (LM) is a one-dimensional chaotic system. The discrete mathematical model of the LM system is defined by (2). The chaotic behaviour appears when $b > 3.569$. Therefore, the initial conditions of the chaotic system must avoid values where the system exhibits periodic-like behaviour.

$$x_{(n+1)} = bx_n(1 - x_n), b \in (0, 4] \quad (2)$$

Related research on chaotic systems has been published. Dridi et al. [27] propose an encryption/decryption procedure operating in cipher block chaining (CBC) mode. The system is produced by a pseudo-random key stream generated by a PRGN based on a chaotic system. Alshammari et al. [28] modified the well-known Advanced Encryption Standard (AES) with a new S-box generated by the Lorenz chaotic map [29]. Zhu et al. [30] propose a cipher schema based on a combined chaotic system between a logistic map and tent map to improve the statistical properties of the generated sequences.

4. Numerical Pseudo-Random Number Generators: Stream ciphers are based on different techniques. The Rivest cipher (RC4) or ARC4 cipher [31] has two subproccesses called the key-scheduling algorithm (KSA) and pseudo-random generation algorithm (PRGA). It is a simple and fast cipher. Several improvements to RC4 have been published. The modified ARC4 (MARC) is one of them [32]. It enhances the security of RC4 by modifying its key-scheduling algorithm (KSA) and improves the performance by modifying the PGRA process. The Salsa20 [33] algorithm was proposed by Daniel J. Bernstein and presented in the eSTREAM project in 2005. It only uses Add–XOR–Rotate operations to generate 512-bit keystreams in each cycle, iterating 20 times the quarter round (QR). The Chacha20 [34] cipher improves upon Salsa20. Chacha20 modifies each word twice in each QR function, while Salsa20 modifies only once per QR.

2.2.2. Block Cipher Algorithms

Block ciphers divide plain data into blocks of the same size that are processed using some of the following functions.

1. Substitution–Permutation Network (SPN): An SPN takes blocks of plaintext and keys and then executes rounds of substitution (S-boxes) and permutation layers to

achieve a ciphertext. The Advanced Encryption Standard (AES) algorithm divides messages into 16-byte blocks with keys of 128, 192, or 256 bits in size. This cipher is an important encryption algorithm due to its high level of security and reduced execution time. Another cipher based on SPN is PRESENT. It was developed with devices with reduced computational resources [35] in mind.

2. Feistel Network (FN): DES [36] is the reference cipher operating in FN balanced mode, with blocks with a 64-bit size and keys of 56 bits during 16 rounds. Its strength resides in a confusion step with 8 S-boxes. However, to break the key, only 2^{56} operations would be necessary. Simon [37] is also based on a balanced FN with n-bit words and a $2 \cdot n$-bit block size. It can operate with different combinations of block sizes ([32–128] bits), key sizes ([64–256] bits), and rounds ([32–72]) to allow more flexibility, bearing in mind constrained devices, where simplicity in design is important.
3. Add–Rotate–XOR (ARX) operations: Speck [37] is a software-orientated cipher and it is based on ARX operations. It can work with different combinations of block sizes ([32–128] bits), key sizes ([64–256] bits), and numbers of rounds ([32–72]).

2.3. Weapon Detection Systems

This section systematically reviews state-of-the-art advancements in weapon detection systems based on deep learning algorithms and highlights how security issues, data transmission protocols, and processing performance are addressed. Then, an analysis of the key contributions and areas for improvement is provided in Table 1.

- A recent review [38] affirms that the Faster R-CNN [39] and You Only Look Once (YOLO) [40] models are architectures that are commonly used. The performance is improved if a dataset with real and synthetic images is used during the training phase. The lighting conditions in images, difficulties in identifying small weapons, lightweight models to use in real-time applications, and unsupervised learning are presented as challenges.
- Another study [41] achieves the objectives of detecting different types of weapons and determining whether the armed human pose is standard or dangerous with high accuracy. In the first phase, the models YOLOv8s, YOLOv8l, and YOLOv8x are implemented. Second, the weighted boxes fusion (WBF) method is used to combine the results obtained from the three YOLOv8 models and to improve the detection accuracy to 30%.
- EfficientDet [42], as an object detection model, is used in [16]. Moreover, the cloud computing paradigm is avoided due to issues such as network latency, data privacy, and slow decision making. Therefore, all processing is performed in the edge layer. A Raspberry Pi 3 is used as an edge device. Positive detections are sent to the server using the Message Queuing Telemetry Transport (MQTT) [43] protocol. This model achieves an inference time of 1.48 s per frame.
- With the objective of improving detection under low light conditions, one study proposes the YOLOv7-DarkVision model [15]. This research applies the techniques of gamma correction, contrast and brightness, Gaussian blur, and normalisation to dark frames. Then, the YOLOv7 model [44] is deployed for the inference phase.
- YOLOv8-AD (You Only Look Once v8—Attack Detection) [45] is a modified version of YOLOv8 [46] that aims to improve the detection of armed soldiers taking attack actions. YOLOV8-AD is based on a dynamic deformable attention mechanism, multi-branch modules with dynamic snake convolution and atrous convolution, a lightweight dynamic detection head with multi-dimensional attention, and a network based on the Inner Minimum Points Distance Intersection over Union (Inner-MPDIoU) loss function.

Experiments were executed on powerful hardware (Nvidia Titan 12GB GPU, Intel Core i7 CPU). The results revealed high precision and strong detection capabilities.

- Another study [47] proposes using YOLOv4 as an object detection model through real-time surveillance cameras. To address the problems of occlusion, hidden handguns, and people close to each other, three heuristics and seven machine learning models are compared. The random forest classifier ML model achieves the best performance.
- The objectives of [48] were to reduce the number of false positives and maintain a low inference time. This was addressed by incorporating a secondary cascaded classifier and a temporal window approach. To enhance the detection of weapons with a small aspect ratio, a scale matching technique was executed. Moreover, a comparative analysis was performed on the YOLOv5, YOLOv7, and YOLOv8 versions for real-time weapon detection. The experiments were executed on a powerful Nvidia A100 with 40 GB of memory.

Table 1. Comparative analysis of weapon detection systems.

Paper (Year)	Key Contributions	Limitations/Areas for Improvement
Systematic review on weapon detection in surveillance footage through deep learning (2024) [38]	• Faster R-CNN and (YOLO) models are architectures commonly used • Train phase using dataset with real and synthetic images improves detection and performance	• Fog computing is not considered • Security issues are not addressed
Multi-Weapon Detection Using Ensemble Learning (2023) [41]	• YOLOv8 model ensembling to increase the accuracy • Focus on light weapon detection	• Fog computing is not considered • Model ensembling may burden edge devices • Security is not addressed • No mention of hardware architecture and resource consumption
Weapons Detection System Based on Edge Computing and Computer Vision (2023) [16]	• Developed Raspberry Pi-based detection system • Edge computing architecture to avoid cloud drawbacks • Positive detections are sent to the server using the MQTT protocol	• Limited resources in the edge layer • Reduced scalability • EfficientDet underperforms vs. YOLOv5 • The MQTT protocol uses the Transport Layer Security [49] protocol, which introduces extra overhead
Robust Weapon Detection in Dark Environments Using YOLOv7-DarkVision (2024) [15]	• Novel approach optimised for low-light conditions • The proposal is based on YOLOv7	• Fog computing is not addressed • YOLOv7 may overload edge devices • Potential data security vulnerabilities during transmission are not addressed • Hardware architecture and resource consumption are not addressed
A Fine-Grained Detection Network Model for Soldier Targets Adopting Attack Action (2024) [45]	• A YOLOv8 improved model is proposed • Targets improved efficiency and robustness in fine-grained soldier detection	• Security considerations for data transmission are not considered • Scalability and fog computing paradigm are not addressed
Improving Armed People Detection on Video Surveillance Through Heuristics and Machine Learning Models (2024) [47]	• Automatic detection of weapons and their carriers • Fusion between YOLOv4, hereuristics, and machine models to resolve occlusions, hidden handguns, and people close to each other	• YOLOv5 is faster and more accurate than YOLOv4 • YOLOv4 is not edge-friendly • Scalability, fog computing, and data security issues are not considered

Table 1. *Cont.*

Paper (Year)	Key Contributions	Limitations/Areas for Improvement
Effective Strategies for Enhancing Real-Time Weapons Detection in Industry (2024) [48]	• Improves real-time detection • Enhances small object accuracy • Reduces false positives	• Requires high-end GPU (RTX-3060) • Edge device limitations are not considered • Security and fog computing are omitted • Scalability deferred to future work

3. Methodology

In this section, we present the proposal, which integrates two main mechanisms. The first one must detect, classify, and possibly identify an armed person. The second must send, protect, and secure these sensitive data. Section 3.1 shows the structure of the data messages. Section 3.2 presents a functional and conceptual description of this proposal.

3.1. Data Block Structure

Data are organised in a data block structure, named the data message of overlapping block. This structure includes a frame check sequence (FCS) for each row and column, as well as sequence number data (SN data). In Figure 2, this structure is presented. Following the scheme proposed in [50], the data block is light-signed so that reply attacks and packet forgery threats can be detected. Figure 2b represents the FCS backwarding method. The FCS included in the latter packets provides a chain-connected mechanism similar to the blockchain one. In the case of corrupted/modified information, the receiver would refuse the message, requesting the resending of affected bytes obfuscated within a block of random data so that no side effect attack can be applied. In this proposal, we select a cyclic redundancy check (CRC-16) for the FCS values, which is a commonly used checksum approach in embedded networks.

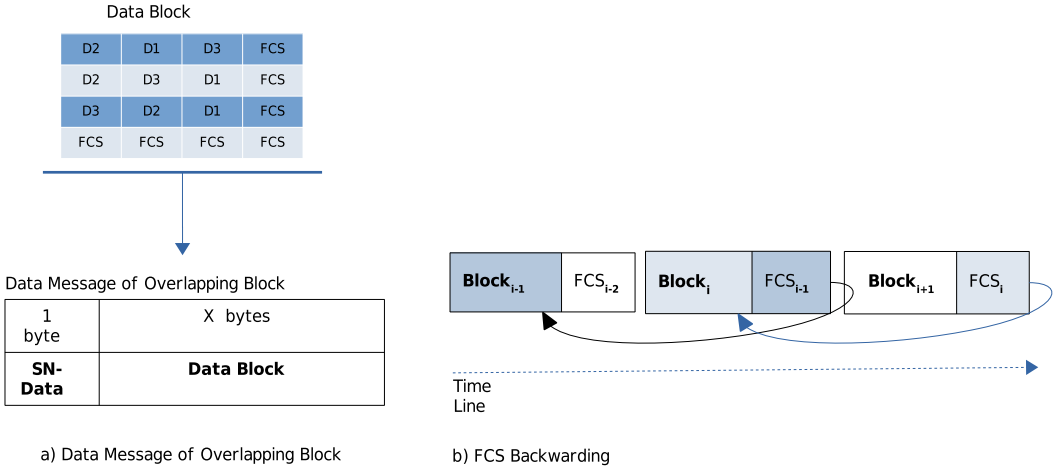


Figure 2. Overlapping block and FCS backwarding.

3.2. Functional Description

The system is structured into edge, fog, and cloud layers. The edge layer includes very low-capacity computing devices that analyse whether there are substantial changes in the scene that warrant processing. When an item of interest appears, it is sent to a GPU in the first fog layer. This GPU analyses whether people and weapons appear. If any appear, the bounding boxes containing the people are selected and sent to an available GPU in the second fog layer. At this layer, the faces of armed individuals are extracted and sent to the

cloud layer for identification. This functionality is exhibited in Figure 3. Data transmitted through pipeline layers are secured by the GS3 mechanism. Sections 3.2.1–3.2.4 describe in detail the method proposed.

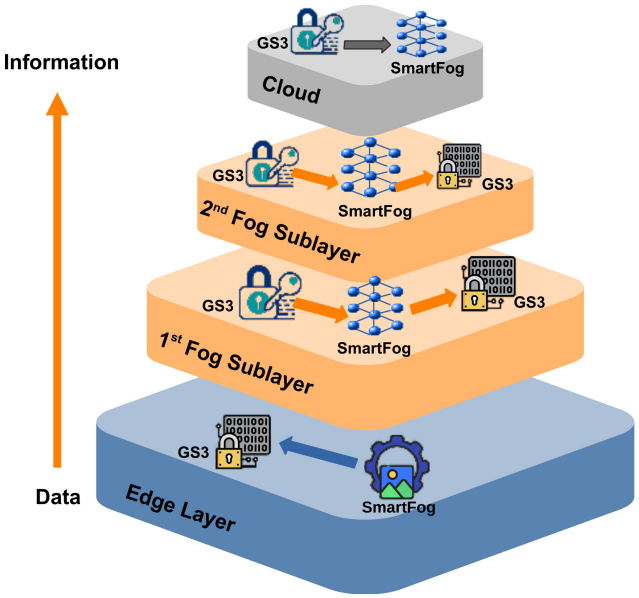


Figure 3. Proposed scheme.

3.2.1. Edge Layer

Figure 4 describes the processes performed in the edge layer. In this part, the SmartFog actor executes the following tasks:

- Capturing an image from a camera;
- Executing the background subtraction (BS) and foreground mask (FM) to decide if there are any changes in the current scene.

Only foreground images exceeding a predefined threshold are taken into account by the GS3 subprocess. GS3 ciphers images by executing the following actions:

- Shuffle subprocess: Firstly, the CRC-16 value for each row and column is calculated. Secondly, a lightweight pseudo-random subprocess to swap the initial positions of the RGB channels of the pixel is applied. This task reaches Shannon's property [51] of diffusion.
- Scramble subprocess: This lightweight method generates a pseudo-random 288-bit string to modify the initial pixel values with an XOR operation. Thus, Shannon's cryptography principle of confusion [51] is accomplished.
- S-box subprocess: The last lightweight subprocess increases the security with a nonlinear transformation, which is based on the S-box substitutions designed by Farah et al. [52], Fatih et al. [53], and Islam et al. [54].

3.2.2. First Fog Sublayer

Figure 5 illustrates the processes executed in the first fog sublayer. In this part, first, the GS3 actor executes the following tasks with the objective of deciphering the received image:

- Inverse S-boxing subprocess;
- Inverse Scrambling subprocess;
- Inverse Shuffling subprocess.

Secondly, SmartFog carries out a DL inference task based on a convolutional neural network. The main objectives are, firstly, to segment the images to obtain the region of interest (ROI) that includes persons and weapons. Next, the bounding box (BB) list is

generated. Each BB is labelled with a probability, size, and coordinates. SmartFog [17] uses the YOLOv5 series [40] as the machine learning model for the detection task. The YOLOv5 series is composed of several different models with a good trade-off between performance and computational requirements, which is of particular importance in low-resources devices in edge and fog layers. Furthermore, SmartFog selects the YOLOv5s subseries [55] to achieve the best trade-off.

Thirdly, GS3 secures only the bounding box list obtained by SmartFog in the previous step. The cipher process is performed by the shuffling, scrambling, and S-boxing subprocesses. These ciphered BBs are sent to the second fog sublayer. The GS3 method could be integrated with any version of YOLO [40] or even with other inference models, such as EfficientDet [42] or YOLO-LSM [56].

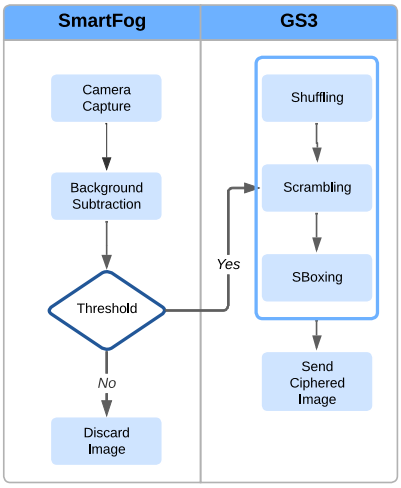


Figure 4. Flow diagram of the edge layer.

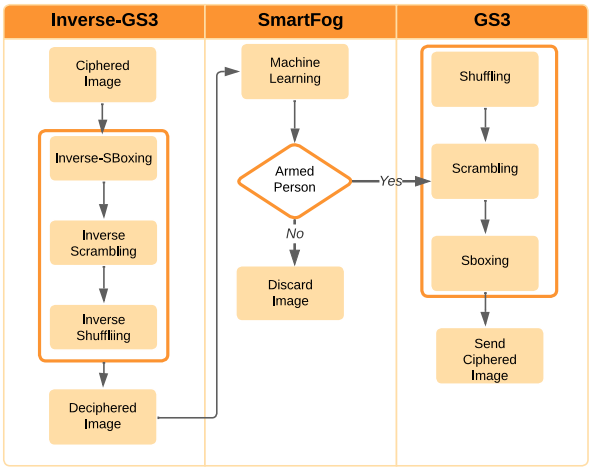


Figure 5. Flow diagram of the first fog sublayer.

3.2.3. Second Fog Sublayer

Figure 6 shows the actors and operations executed by them. Firstly, GS3 must decipher the received images from the first fog sublayer through inverse shuffling, scrambling, and S-boxing subprocesses. Secondly, the SmartFog actor tries to detect faces using the YOLOv5s6 model. Thirdly, these detected faces corresponding to armed persons are ciphered by the GS3 actor. At other times, the shuffling, scrambling, and S-boxing subprocesses are responsible for ciphering.

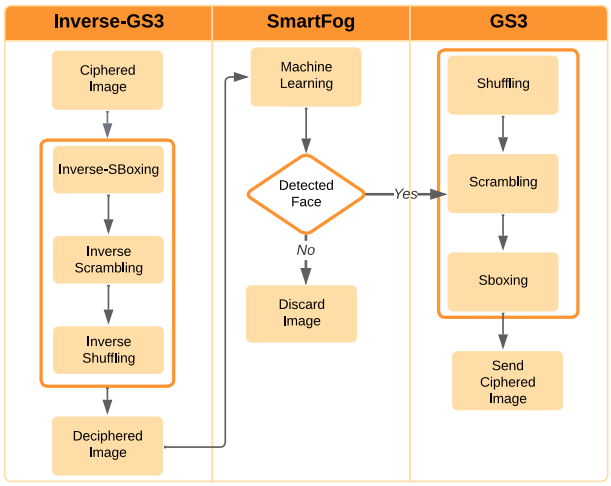


Figure 6. Flow diagram of the second fog sublayer.

3.2.4. Cloud Layer

When the images reach the cloud layer first, GS3 deciphers the images following the inverse shuffling, scrambling, and S-boxing subprocesses. Second, the clear images are analysed to identify faces using the Python library [57]. Moreover, if the identified face has no permission to carry a weapon, an alarm could be raised. Figure 7 shows these operations.

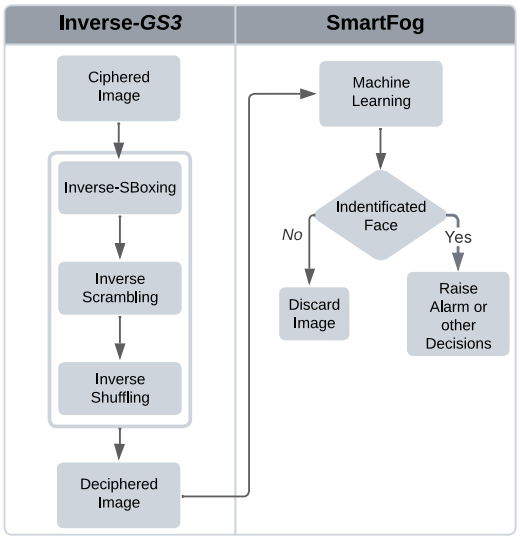


Figure 7. Cloud layer flow diagram.

3.3. Deployment Diagram

Figure 8 presents the deployment diagram of the proposed mechanism. An example of the hardware in each layer could be the following.

- Edge layer: This layer is composed of grid sensor platforms with limited computational resources—for example, Raspberry Pi platforms with a camera for capturing images.
- First or second fog sublayer: Network-enabled platforms with moderate computational resources—for example, Nvidia Jetson Nano devices equipped with a graphics processing unit (GPU) to execute machine learning processes.
- Cloud layer: This layer is composed of devices with high-end resources for the CPU, GPU, and memory—for example, an AMD Ryzen server with a GeForce Nvidia GPU.

The raw data are driven up from the edge layer to the cloud layer. Moreover, the nodes in the fog sublayers could accept connections from multiple nodes of the preceding fog sublayer.

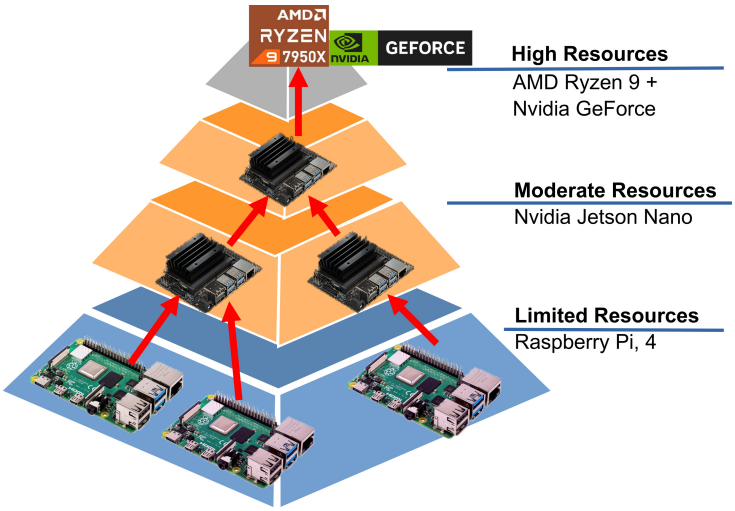


Figure 8. Deployment diagram.

4. Implementation

This section describes the hardware and software used to assess our proposal.

- Edge layer
  - Hardware: Raspberry Pi 4 Model B Rev 1.4. ARMv7l @ 1.8 GHz, 2 GB RAM.
  - Software: Raspbian GPU/Linux 10 (buster) operating system. C++ programming language.
- First and second fog sublayers
  - Hardware: Nvidia Jetson Nano with ARMv8 Processor, 4 Cores @ 1.4 GHz, 2–4 GB RAM, and Nvidia Maxwell GPU.
  - Software: Nvidia Jetson Nano, Jetpack 4.6. C++ and Python are programming languages.
- Cloud layer
  - Hardware: AMD Ryzen 9 computer, with an Nvidia GeForce RTX 2070 GPU.
  - Software: Debian 12 as the operating system. C++ and Python programming languages.

5. Experimentation

This section describes the experiments used to assess and validate the proposed method. The main objective is to measure the overhead introduced by the GS3 method [11]. Therefore, the parameters analysed are related to the computational resource consumption. The measurements consider the following:

- Evaluation of the reviewed lightweight cipher algorithms;
- Execution time;
- CPU percentage consumption;
- GPU percentage consumption;
- Memory percentage consumption;
- Power consumption.

These parameters are measured in each fog sublayer. Firstly, the SmartFog mechanism is executed alone, and then the GS3 method is run, integrated with SmartFog [17]. Moreover, we provide a broader analysis to support our proposal, including a consumption comparison between the GS3 method and the Advanced Encryption Standard (AES) algorithm [18]. To maintain the same conditions in all experiments, the web camera of the

Raspberry Pi device is replaced with a sequence of videos. This sequence [58] represents five different scenarios, which are the following.

1. Background (B): No person appears in this video. The edge node tries to detect variations in the background scene. Therefore, the percentage of data sent to the first fog sublayer will be negligible or zero.
2. Person (P): Scenes are composed of one person walking or standing. This person does not carry any weapon. However, it is expected to have some data traffic between the edge and the first fog sublayer. Therefore, secure data communications are sent.
3. People (Pn): This scenario is similar to the prior case. However, there are several people involved who are walking or standing. All data communications to the first fog sublayer are protected. Nevertheless, no traffic is expected to the second fog sublayer because no weapon is detected.
4. Armed person (AP): This video contains a person holding a weapon. With all communications secured, this scenario will send data through the first and second fog sublayers.
5. Armed people (APn): This video contains scenes with several people holding guns in the same frame. Moreover, a person can hold different weapons at the same time. It is expected that a higher number of secured communications will occur between fog sublayers.

5.1. Evaluation of the Reviewed Lightweight Cipher Algorithms

This section presents a quantitative and qualitative evaluation to concisely review the cipher mechanisms evaluated to develop the GS3 method [11]. First, Section 5.1.1 defines the security metrics evaluated. Second, Section 5.1.2 shows the scores achieved for each metric.

5.1.1. Security Metrics

The reviewed ciphers are evaluated with the following parameters.

- Execution time: This parameter is commonly used to compare the consumption of resources in different algorithms. However, it is crucial in resource-constrained IoT devices.
- Entropy test: This value measures the randomness of the ciphered data. Therefore, the entropy value is at the maximum if all values of the data have the same or a similar probability [51]. Entropy is defined by Equation (3):

$$H(S) = -\sum_{i=1}^{N} P(s_i) log_2 P(s_i) \tag{3}$$

  The S variable is random and it produces output values within $(s_1, \ldots, s_N)$, and $P(s_i)$ is the probability of occurrence of the s_i output.
- χ^2 test: This metric is used to confirm the results of the entropy test. The higher the uniformity of the ciphered data is, the harder it is for an intruder to obtain helpful knowledge about the data. The χ^2 test is defined by Equation (4). Considering 8 bits per pixel, $k = 2^8$, o_i is the observed probability and e_i is the expected probability of the grey level k_i, respectively.

$$X^2 = \sum_{i=1}^{k} \frac{(o_i - e_i)^2}{e_i} \tag{4}$$

- Key Space: The total number of distinct keys that can be generated to encrypt the data. The recommendation [59] for a minimal-length key for symmetric algorithms is 112 bits.

5.1.2. Scores of the Security Metrics

Table 2 shows the scores achieved for each metric, taking into account the following considerations.

- These measures were taken on a Raspberry Pi model B, ARM @ 1.8 GHz and 2 GB RAM devices, using Python as the programming language.
- The image used for the test was the well-known Lake [60] image in its greyscale version, with a size of 256 × 256 bytes and an 8-bit resolution.
- Execution time score: We set a maximum value of two seconds. The achieved score is calculated with (5):

$$Score_T = 1 - \frac{Execution_Time}{2} \tag{5}$$

- Entropy score: Considering an 8-bit resolution, the ideal value of entropy is 8. This value is provided by (6):

$$Score_E = \frac{Entropy}{8} \tag{6}$$

- χ^2 score: Considering a significance level of $\alpha = 0.05$, the value is obtained by (7). According to the 255-degrees-of-freedom χ^2 table and $\alpha = 0.05$, the critical value of χ^2 is $\chi^2 = 293.247$. Therefore, this test is passed if $\chi^2 < 293.247$.

$$Score_{\chi^2} = 1 - \frac{\chi^2}{293.24} \tag{7}$$

- Key space: The key space score is calculated with (8). The maximum number of keys in this case is 544.

$$Score_K = \frac{KeySpace}{544} \tag{8}$$

- Sum score: The end score is the sum of each achieved mark, which is defined in (9):

$$Sum_Score = \sum_{i=\{T,E,\chi^2,K\}} Score_i \tag{9}$$

In Figure 9, the marks are represented on a logarithmic scale.

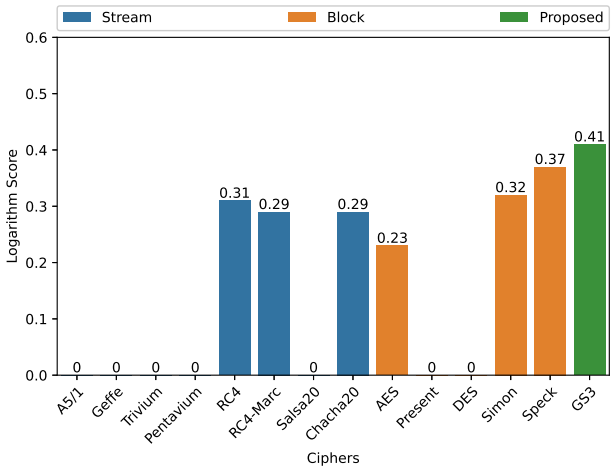


Figure 9. Score bar diagram.

5.1.3. Analysis of Security Strengths

This section aims to evaluate the security robustness of GS3, executing the proofs described in the following sections.

Table 2. Measures and scores of the cipher algorithms.

	Category	Cipher	Time (s)	Score	Entropy	Score	χ^2	Score	Key Space	Score
Stream Ciphers	LFSR	A5/1 [23]	48.8	−23.4	7.99	0.99	248	0.15	64	0.11
		Geffe [24]	700	−349	7.99	0.99	238	0.18	64	0.11
	Celular Automata	Pentavium [26]	443	−220	7.99	0.99	260	0.11	160	0.29
		Trivium [61]	15.2	−6.6	7.99	0.99	291	0.007	160	0.29
	Numerical	RC4 [31]	1.04	0.48	7.98	0.99	198	0.32	256	0.47
		RC4-Marc [32]	0.75	0.62	7.99	0.99	264	0.09	256	0.47
		Salsa20 [33]	47.2	−22.6	7.99	0.99	252	0.13	512	0.94
		Chacha20 [34]	1.95	0.027	7.99	0.99	648	−1.2	512	0.94
Block Ciphers	SPN	AES [18]	1.7	0.11	7.99	0.99	256	0.12	256	0.47
		Present [35]	18.3	−8.15	7.99	0.99	299	−0.01	256	0.47
	FN	DES [36]	24.3	−11.1	7.99	0.99	259	0.11	256	0.47
		Simon [37]	0.85	0.57	7.99	0.99	270	0.07	256	0.47
	ARX	Speck [37]	0.57	0.71	7.99	0.99	256	0.12	256	0.47
Proposal		GS3 [11]	1.21	0.39	7.99	0.99	242	0.17	544	1.00

Simulated Attack Model

This section analyses the capacity of GS3 to detect the following attacks.

1. Tampering packet: A malicious node alters some data packets and then forwards them.
2. Replay attack: An intruder resends old data packets. Therefore, the freshness of the data is affected.
3. Packet forgery: A malicious node sends fake data packets, so the traffic network is increased and the energy consumption rises consequently.
4. Selective forwarding: A malicious node sends data packets selectively, and others can be partially deleted. Therefore, this may cause a loss of data.

Each experiment was repeated 50 times, and 10 data blocks were sent, each with a 148-byte length. Table 3 shows percentage of detection for each type of attack.

Table 3. Percentage of detection.

Type of Attack	Attacks	Detected
Tamper	355	100%
Replaying	435	100%
Forwarding	420	100%
Forgery	390	100%

Analysis of the Key Space

The strength of the proposed method should be analysed by considering the three subprocesses working together. In the case of a known plaintext attack (KPA), the intruder should compare each cipher datum generated with the known cipher data. First, 64 bits of precision are considered for the b and x_n parameters defined in Equation (2). Therefore, each logistic map achieves 2^{128} possible permutations. The shuffle process reaches 2^{128} possible permutations; the scramble process, with an internal state of 288 bits, achieves 2^{288}; and the S-box process delivers 2^{128} permutations. Thus, the global key space, defined in Equation (10), is large enough so that it is unfeasible to execute zero-knowledge (brute force) attacks.

$$Permutations = 2^{128} \cdot 2^{288} \cdot 2^{128} = 2^{544} \tag{10}$$

Differential Cryptoanalysis

This section studies the strength of GS3 against attacks based on differential analysis. The objective is that small changes in plain data or private keys must provoke greatly different ciphered data. The number of pixels change rate (NPCR) and unified average changing intensity (UACI) metrics measure this aspect [62]. The optimum values of these parameters are 99.61% and 33.46% for the NPCR and UACI, respectively. The test run was the following:

1. Select a byte randomly from the plaintext and modify the four most significant bits (MSb) of it. When altering only 1 bit, the values of the NPCR and UACI were 99.607 and 33.511, respectively.
2. The same data block was scrambled with small changes in the seeds or keys. When altering only the most significant bit of the parameter b in Equation (2), the values of the NPCR and UACI were 99.61 and 33.67, respectively.

Randomness Tests

This section evaluates the randomness of the generated sequence by the scrambling process through the following tests.

1. Ent test suite [63]: This test was run with a binary file of about 10^6 bits. The results were as follows: entropy, 1.00 bits per bit; arithmetic mean value of data bits, 0.50, where 0.5 means random; Monte Carlo value for Pi, 3.138799615; and serial correlation coefficient, 0.000069.
2. 800-22-rev1 test suite [64]: This was run with 100 sequences of 100,000 bits. Only 2 out of the 15 tests that compose the suite fail to assess the randomness, specifically the *dft* and *overlapping-template-matching* tests.

5.2. Edge Layer

This section describes the experiments performed in the edge layer. These measurements were conducted on a Raspberry Pi device.

5.2.1. Cipher Execution Time of GS3 Method in the Edge Layer

Figure 10 presents the execution times of the GS3 method in seconds. In detail, the CRC, shuffling, s-boxing, scrambling, and the average sum of these times are shown for each sent frame. The average time for ciphering is 0.059 s.

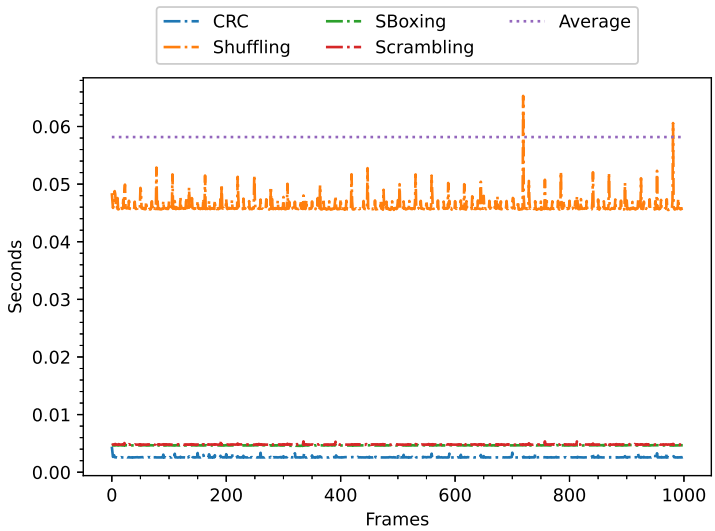


Figure 10. Ciphering execution times of GS3 in the edge layer.

5.2.2. Cipher Execution Time of AES Algorithm in the Edge Layer

This section shows the execution time of the AES algorithm at the edge layer in seconds. Figure 11 provides these times and the average values of these times for each sent frame. The average time of ciphering is 0.0966 s.

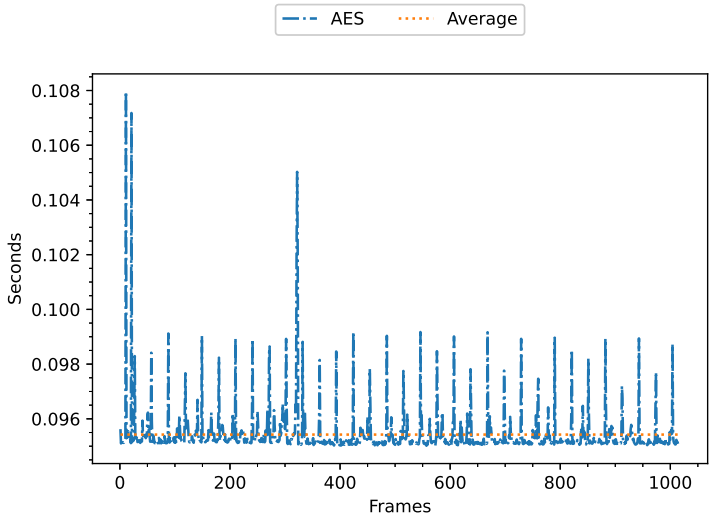


Figure 11. Ciphering execution time of AES algorithm in the edge layer.

5.2.3. Execution Time of SmartFog Method in the Edge Layer

This section shows the execution time (in seconds) of the SmartFog method at the edge layer. These execution times are due to the overall use of the background subtraction (BS) and foreground mask (FM) techniques. Figure 12 shows these times and the average values of these times for each sent frame, which spanned 0.0016 s.

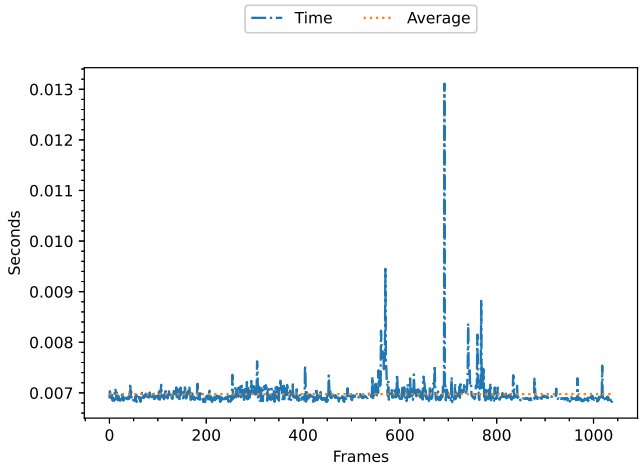


Figure 12. Execution time of SmartFog in the edge layer.

5.2.4. Quantitative Results of Execution Time, CPU, and Memory Consumption in the Edge Layer

Table 4 This section a shows the average values of the cipher execution time for the GS3 and AES algorithms and the execution time of the SmartFog method. Moreover, the average percentages of CPU, GPU, and memory consumption were calculated during the interval time to send all frames from the edge layer. These measures were sampled every three seconds. Figure 13 presents the average percentage of overhead in resource consumption caused by the secure methods. The percentage of resource consumption by the SmartFog method is represented in blue, and orange denotes GS3 and AES. The blue bar

represents the percentage of overhead of the SmartFog part, while the orange bar represents the percentage of overhead for the secure method.

Table 4. Quantitative results in the edge layer.

Method	Operation	$\overline{Time(\mathrm{s})}$	$\overline{\%CPU}$	$\overline{\%RAM}$
GS3	Cipher	**0.058**	–	–
AES	Cipher	0.095	–	–
SmartFog	BS	0.0056	10.24	2.54
SmartFog + AES	Combined	0.102	44.65	7.42
SmartFog + GS3	Combined	**0.063**	**34.85**	**5.21**

Note: Bold values indicate the lowest overhead for each parameter. BS: background subtraction. Combined: GS3 and SmartFog operations, such as cipher/decipher and inference.

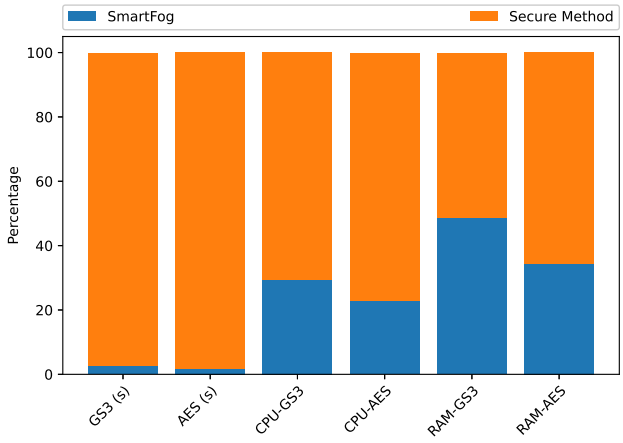


Figure 13. Percentage of overhead introduced by GS3 and AES in the edge layer.

5.3. First Fog Sublayer

This section describes the measures carried out in the first fog sublayer. These measures are shown for Nvidia Jetson Nano devices.

5.3.1. Decipher Execution Time of GS3 Method in the First Fog Sublayer

Figure 14 presents the execution times of the GS3 method in seconds. In detail, the CRC and the inverse processes of shuffling, S-boxing, and scrambling are shown. Additionally, the total average execution time is displayed. The average deciphered value is 0.03 s.

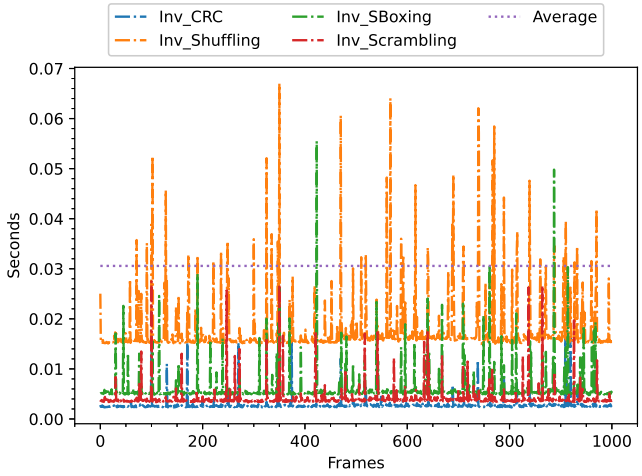


Figure 14. Decipher execution time of GS3 method in the first fog sublayer.

5.3.2. Cipher Execution Time of GS3 Method in the First Fog Sublayer

The GS3 method must cipher the frames detected by the neuronal network. Figure 15 shows the execution times required to calculate the CRC-16 values and execute the Shuffling, S-boxing, and Scrambling processes. The average time for ciphering is 0.013 s.

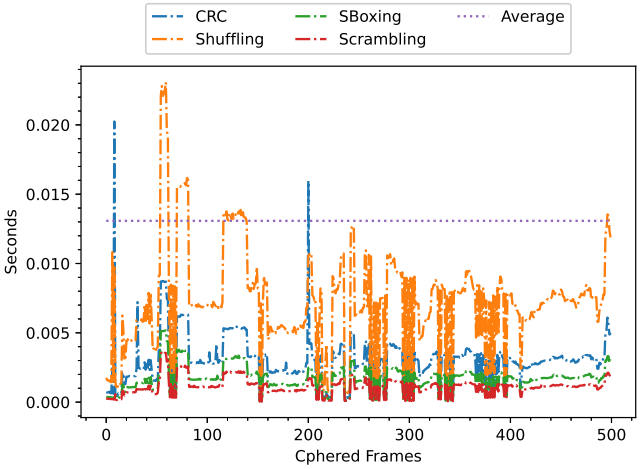


Figure 15. Ciphered execution time of GS3 mechanism in the first fog sublayer.

5.3.3. Inference Time in the First Fog Sublayer

The inference time is the required processing time for the neuronal network to evaluate and classify the received frames. Figure 16 shows these times, with an average time of 0.253 s per frame.

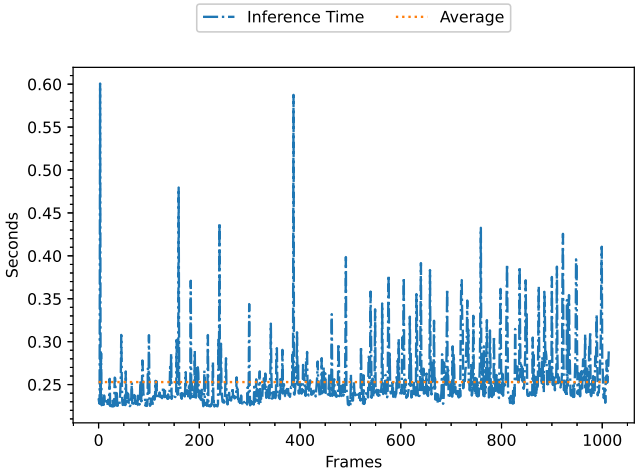


Figure 16. Inference time in the first fog sublayer.

5.3.4. Decipher Execution Time of AES Algorithm in the First Fog Sublayer

Figure 17 shows the execution times taken to decipher the frames with the AES algorithm. The average deciphering time is 0.1 s.

5.3.5. Cipher Execution Time of AES Algorithm in the First Fog Sublayer

Figure 18 shows the execution times taken to cipher the frames with the AES algorithm. The average ciphering time is 0.029 s.

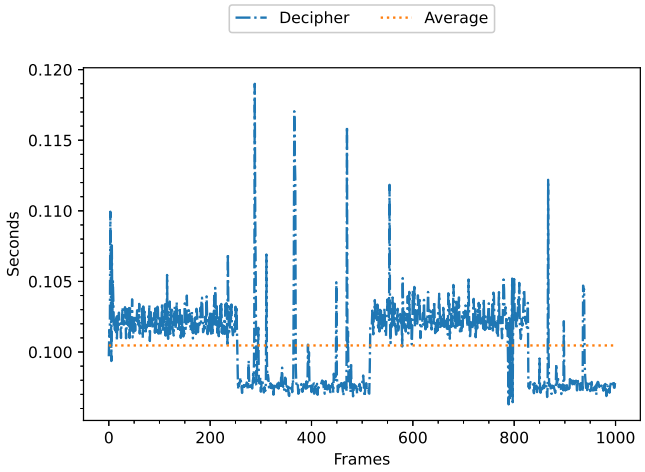


Figure 17. Decipher time of AES algorithm in the first fog sublayer.

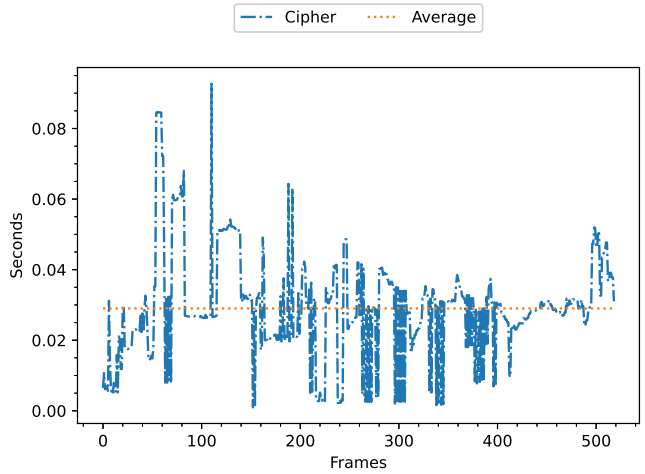


Figure 18. Cipher time of AES algorithm in the first fog sublayer.

5.3.6. Quantitative Results of Execution Time, CPU, GPU, and Memory Consumption in the First Fog Sublayer

This section shows in Table 5 the average values of the execution time, CPU, GPU, and memory consumption in the Nvidia Jetson Nano device for the GS3, SmartFog, and AES algorithms. The average percentages of CPU, GPU, and memory consumption were calculated during the interval time to receive all frames from the edge layer. These measures were sampled every three seconds. Moreover, the introduced overhead for the GS3 and AES mechanisms concerning the SmartFog method is shown in Figure 19. In boldface, the lowest overhead is indicated for each case.

Table 5. Quantitative results in the first fog sublayer.

Method	Operation	$\overline{Time(\mathrm{s})}$	$\overline{\%CPU}$	$\overline{GPU}$	$\overline{\%RAM}$
GS3	Decipher	**0.030**	–	–	–
	Cipher	**0.013**	–	–	–
AES	Decipher	0.100	–	–	–
	Cipher	0.029	–	–	–
SmartFog	Inference	0.253	82.83	62.10	76.19
SmartFog + AES	Combined	0.382	93.01	60.37	93.85
SmartFog + GS3	Combined	**0.296**	**89.59**	**55.84**	**92.06**

Note: Bold values indicate the lowest overhead for each parameter. Combined: GS3 and SmartFog operations, such as cipher/decipher and inference.

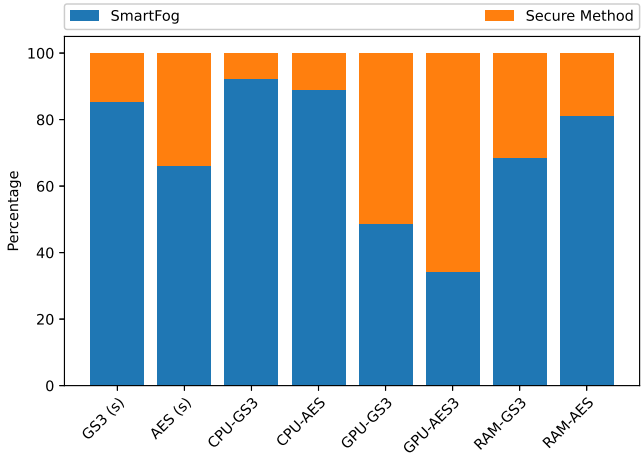


Figure 19. Percentage of overhead introduced by GS3 and AES in the first fog sublayer.

5.4. Second Fog Sublayer

This section describes the measures carried out in the second fog sublayer. These measures are provided for Nvidia Jetson Nano devices.

5.4.1. Decipher Execution Time of GS3 Proposal in the Second Fog Sublayer

Figure 20 presents the execution times of the GS3 method in seconds. In detail, the CRC and the inverse processes of Shuffling, S-boxing, and Scrambling are shown. Moreover, the total average execution time is shown. The average deciphering time is 0.0106 s.

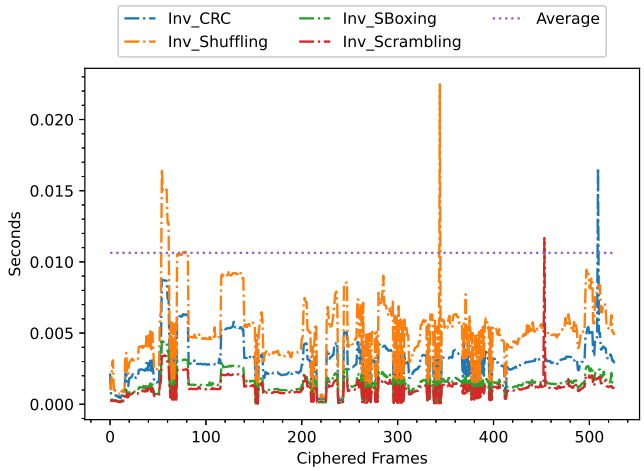


Figure 20. Decipher execution time of GS3 proposal in the second fog sublayer.

5.4.2. Cipher Execution Time of GS3 Proposal in the Second Fog Sublayer

The GS3 method must cipher the frames detected by the neuronal network. Figure 21 shows the execution times taken to calculate the CRC-16 values and execute the shuffling, S-boxing, and scrambling processes. The average ciphered value is 0.00103 s.

5.4.3. Inference Time in the Second Fog Sublayer

The inference time is the required processing time for the neuronal network to evaluate and classify the received frames from the first fog sublayer. Figure 22 shows these times, with an average time of 0.202 s for each frame.

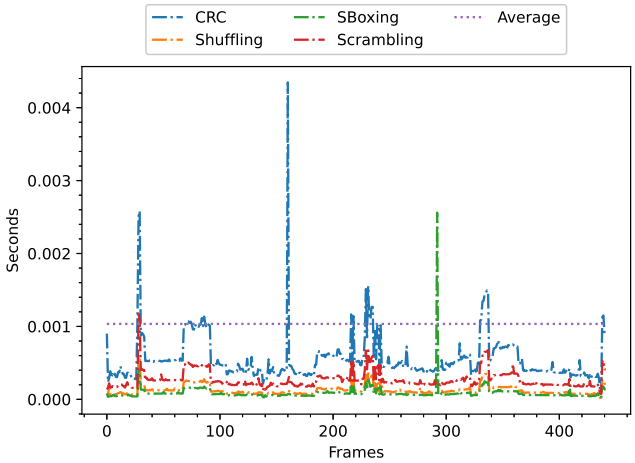


Figure 21. Cipher execution time of GS3 proposal in the second fog sublayer.

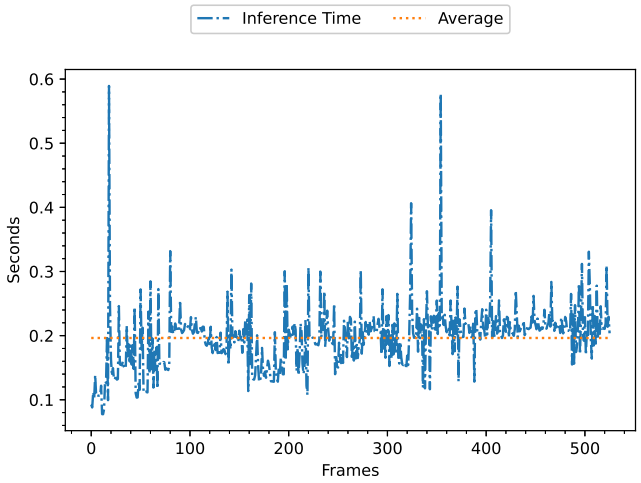


Figure 22. Inference time in the second fog sublayer.

5.4.4. Decipher Execution Time of AES Algorithm in the Second Fog Sublayer

Figure 23 shows the execution times taken to decipher the frames with the AES algorithm. The average deciphered value is 0.025 s.

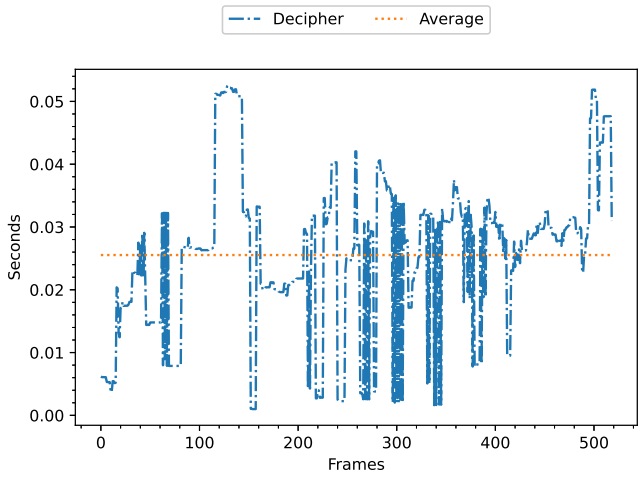


Figure 23. Decipher execution time of AES algorithm in the second fog sublayer.

5.4.5. Cipher Execution Time of AES Algorithm in the Second Fog Sublayer

Figure 24 shows the execution times taken to cipher the frames with the AES algorithm. The average ciphering time is 0.00202 s.

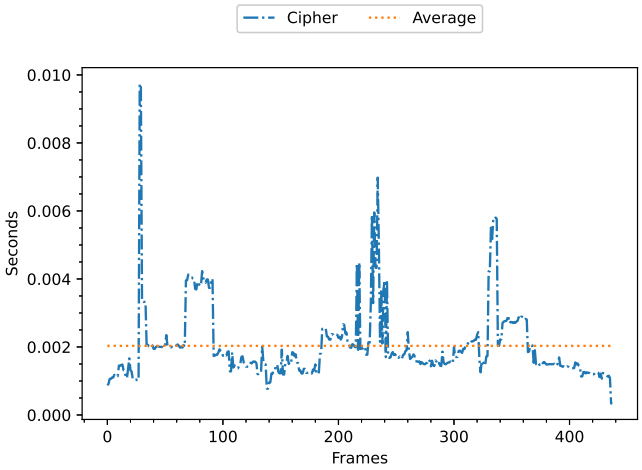


Figure 24. Cipher execution time of the AES algorithm in the second fog sublayer.

5.4.6. Quantitative Results of Execution Time, CPU, GPU, and Memory Consumption in the Second Fog Sublayer

Table 6 shows the average values of the execution time, CPU, GPU, and memory consumption in the Nvidia Jetson Nano device for the GS3, SmartFog, and AES algorithms. The average percentage of CPU and memory consumption was calculated during the interval time to receive all frames from the first fog sublayer. These measures were sampled every three seconds. Moreover, the introduced overhead for the GS3 and AES mechanisms with respect to the SmartFog method is shown in Figure 25.

Table 6. Quantitative results in the second fog sublayer.

Method	Operation	$\overline{Time(\mathrm{s})}$	$\overline{\%CPU}$	$\overline{\%GPU}$	$\overline{\%RAM}$
GS3	Decipher	**0.011**	–	–	–
	Cipher	**0.001**	–	–	–
AES	Decipher	0.025	–	–	–
	Cipher	0.002	–	–	–
SmartFog	Inference	0.196	64.8	21.31	92.44
SmartFog + AES	Combined	0.229	70.95	9.73	92.78
SmartFog + GS3	Combined	**0.213**	**68.14**	**9.11**	**91.96**

Note: Bold values indicate the lowest overhead for each parameter. Combined: GS3 and SmartFog operations, such as cipher/decipher and inference.

5.5. Power Consumption

Energy consumption is critical in sensor networks [65]. We estimated the energy consumption of our proposal and compared it with that of the popular AES encryption method [18]. The best layer in which to take these measures is the edge layer because it is not affected by communications, the inference process, or pipeline stalls in devices. Figure 4 shows the processes involved, while Figure 26 presents the measurement elements, which include the following:

- The Agilent 3631A power supply [66].
- A Raspberry Pi [67] as an edge device; it is powered by a regulated Agilent 3631A power source.

- The Agilent 34405A multimeter to measure the current; it is connected in series with the circuit.
- The Agilent 34405A multimeter is connected to a computer via a USB port to transfer current samples using the Virtual Instrument Software Architecture (VISA) protocol [68] developed by IVI Foundation [69]. A Python script is run on the computer that receives the measures. These are interpreted by the PyVISA library [70]. After this, the Pandas Python library [71] allows the reading and analysis of the current measures. Then, data are represented using the Matplotlib Python library [72].

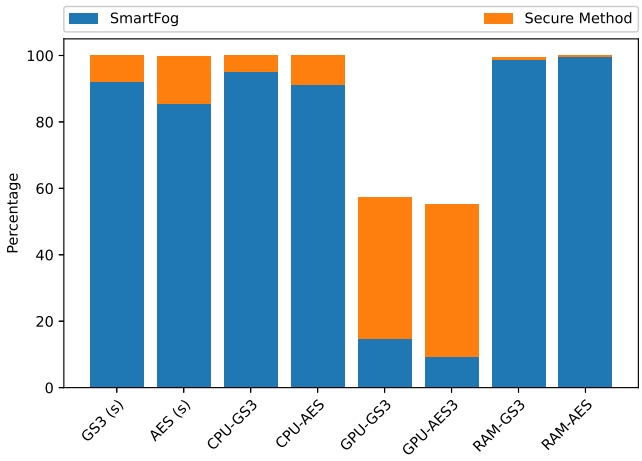


Figure 25. Percentage of overhead introduced by GS3 and AES in the second fog sublayer.

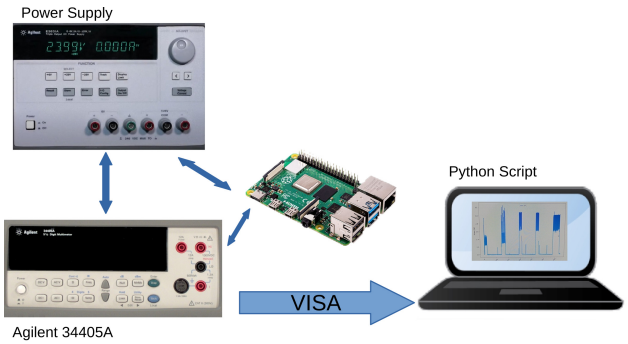


Figure 26. Diagram of power consumption measurement.

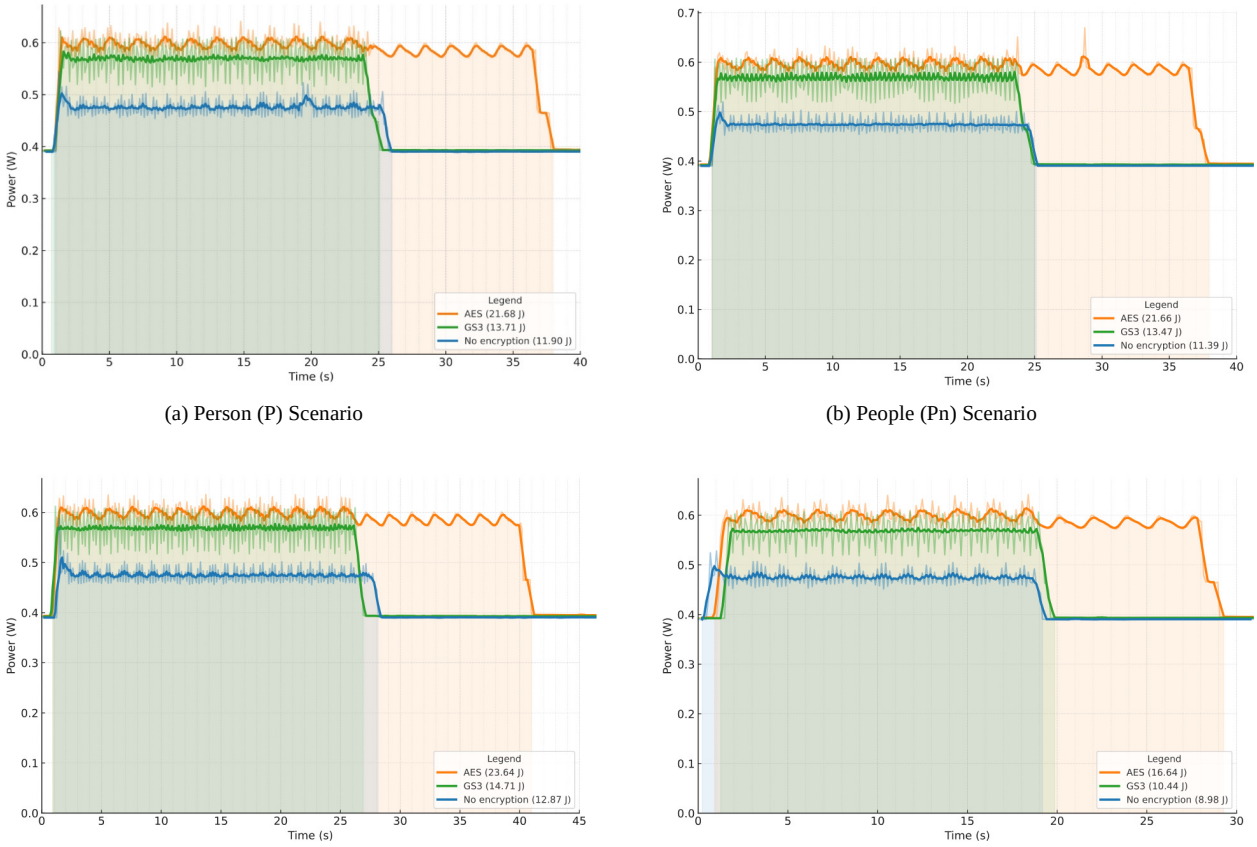
Secondly, the sequence of videos (B, P, Pn, AP, APn) is sent from the edge layer to the first fog sublayer. First, each scenario is sent without securing communications. Second, the GS3 method is employed to secure communications. Third, the AES algorithm is executed to cipher the data. Figure 27 exhibits the power consumption required to send and cipher the videos corresponding to the scenarios (P, Pn, AP, and APn, respectively) from the edge device. The (B) scenario produces no send; for this reason, this scenario is not shown in Figure 27. Blue highlights the baseline power draw of unsecured frame processing and transmission. Orange and green represent the power consumption required to process and transmit encrypted frames with the AES and GS3 methods, respectively. The energy consumption is proportional to the area under the respective curve.

5.6. Visual Results

We include this section to achieve the following visual objectives:

1. Figure 28a,b present ciphered frames with the GS3 proposal and AES algorithm, respectively.

2. After the deciphering process, the received frame in the first fog sublayer is processed and analysed by our weapon detection system. As a result, bounding boxes identifying armed persons are computed. Figure 29a exhibits this processing.
3. After deciphering the received frame in the second fog sublayer, bounding boxes for the detection of faces are generated. Figure 29b shows the detected faces. These faces will be recognised in the cloud layer.



(a) Person (P) Scenario

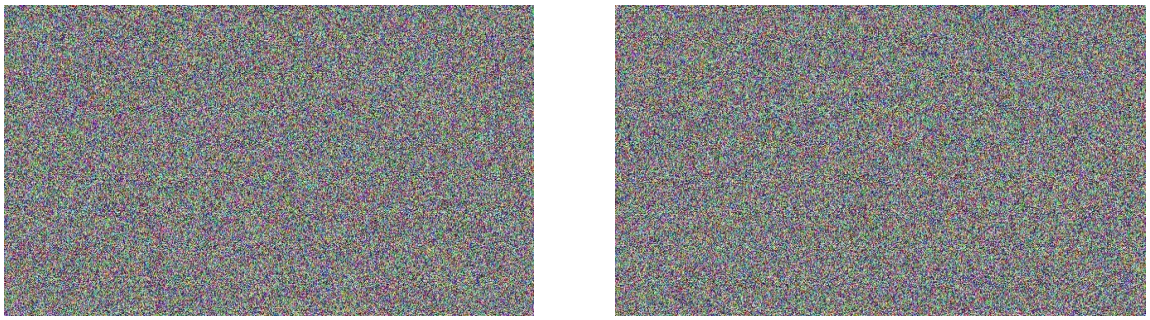
(b) People (Pn) Scenario

(c) Armed Person (AP) Scenario

(d) Armed People (APn) Scenario

Figure 27. Power consumption measures.

The videos used in these experiments are licensed under the Creative Commons Zero (CC0) license [58].



(**a**) (**b**)

Figure 28. (**a**) Ciphered frame in the edge layer with the GS3 proposal. (**b**) Ciphered frame in the edge layer with the AES algorithm.



(**a**) (**b**)

Figure 29. (**a**) Predicted bounding boxes in the first fog sublayer. (**b**) Predicted bounding boxes in the second fog sublayer.

6. Discussion

This section provides an in-depth analysis and comparison of the experimental results presented in Section 5. Firstly, Section 5.1 presented a classification and evaluation of the reviewed lightweight cipher algorithms used to design the GS3 method [11]. Table 2 shows the results of this evaluation. The Simon and Speck algorithms, described in Section 2.2.2, achieved the lowest execution times, with 0.85 and 0.57 s, respectively. The GS3 method performs in 1.21 s, about 30% less than the AES algorithm; see Section 2.2.2. The Pentavium, Trivium, Salsa20, and PRESENT algorithms (see Section 2.2.1) delay the execution time. Although RC4 (see Section 2.2.1) achieves an execution time of 1.04 s, this algorithm does not consider the important principle of diffusion in Shannon's cryptography [51]. On the other hand, GS3 addresses this principle. Furthermore, RC4 is known to suffer from statistical and brute force attacks; for these reasons, RC4 was prohibited in Transport Layer Security by RFC 7465.

With respect to the security metrics described in Section 5.1.1 and the entropy test, all ciphers pass the test, achieving a value close to 8. When analysing the results of the χ^2 test, only Chacha20 and Present fail this test. These metrics are combined to calculate a global score, defined in Section 5.1.2. Figure 9 displays these scores, where GS3 exhibits a 0.41 score.

In Section 5.1.3, the security strength of GS3 is examined from four perspectives. Firstly, as described in Section 5.1.3, 100% of the tampering, replaying, forwarding, and forgery attack methods were detected, and these blocks were refused, as Table 3 shows. Secondly, the analysis of the key space in Section 5.1.3 reveals a global key space of 2^{544} permutations, which is followed by the Salsa20 and Chacha20 algorithms. This space is much larger than in the Simon and Speck algorithms; although GS3 is slower than Simon or Speck, by striking a balance between the key space length and speed, GS3 is a much stronger solution. In addition, GS3 passes the entropy test and "800-22-rev1" randomness test presented in Section 5.1.3. Furthermore, the security strength against plaintext attacks, as presented in Section 5.1.3, shows that GS3 passes the NPCR and UACI tests.

Secondly, Sections 5.2–5.4 consider the impact of introducing a secure method in the SmartFog proposal [17]. In Figures 10 and 11 and Table 4, we show the cipher execution time in the edge layer. GS3 requires 0.058 s, and AES performs in 0.095 s, which is about

38% greater than that of GS3. Nevertheless, both securing methods increase the execution time with respect to the SmartFog mechanism. Figure 12 displays the execution time of SmartFog. This occurs because SmartFog performs lightweight operations with minimal branching and memory use at the edge. As a consequence, GS3 and AES show larger memory and CPU footprints than SmartFog. Figure 13 shows the percentage of overhead introduced by both secure methods. The results in Table 4 demonstrate that GS3 requires fewer computational resources than AES.

In the first fog sublayer, on the one hand, a comparison between Figures 13 and 19 shows that SmartFog consumes the most computational resources. This consumption stems directly from the inference process, as detailed in Figure 16. On the other hand, as evidenced by Figures 14, 15, 17 and 18, GS3 outperforms AES because it reduces the decipher time by 70% and the cipher time by 55%, respectively. Nevertheless, both secure methods entail a longer execution time and higher CPU and memory consumption, as demonstrated in Figure 19 and Table 5. GS3 leads to a 14% increase in execution time; however, AES results in a 34.5% longer execution time. With respect to memory consumption, both algorithms increase this by around 18%. An important observation when comparing the number of frames processed by the inference phase (Figure 16) and the frames that are ciphered and then sent (Figure 15) is that only the bounding boxes of armed people from the first to the second fog sublayer are sent. This behaviour is evidenced in Figure 29a,b. Therefore, the data transmission volume is dramatically reduced along the fog layers. Furthermore, the cipher/decipher execution time decreases from the edge layer to the cloud layer because the frames are segmented by bounding boxes (BB), where armed persons appear in the first fog sublayer and then only faces in the second fog sublayer.

Regarding the second fog sublayer, in Figures 20 and 23, the decipher execution time is displayed in detail for GS3 and AES. Once again, GS3 shows superior performance with respect to AES, with around 57% less time consumed than AES. Similar behaviour for the cipher execution time is shown in Figures 21 and 24, which display the cipher execution time per face detected for GS3 and AES, respectively. GS3 requires around 50% less. These results lead to an 8% increase in the global processing time with GS3, but that of AES increases by 14%. Note that, in the second fog sublayer, the GS3 mechanism slightly reduces the memory consumption due to the reduced data volume after face detection filtering. Table 6 summarises these measurements.

Regarding power consumption (Section 5.5), the standby power consumption is close to 0.4 watts. When analysing the energy consumption of the Pn scenario in Figure 27b, no encryption frame processing and transmission require 11.39 Joules. Securing frames with the GS3 method increases the energy consumption to 13.47 Joules. This introduces a 15% overhead in energy usage. However, AES raises the energy consumption to 21.66 Joules. This constitutes a 46% increase in consumption. Therefore, GS3 leads to 31% less power consumed. Similar behaviour is exhibited in the other scenarios (P, AP, and APn) in Figure 27a, Figure 27c, and Figure 27d, respectively.

The visual result after ciphering the frame shown in Figure 29a with GS3 and AES is depicted in Figure 28a and Figure 28b, respectively. As demonstrated, both methods produce cipher images exhibiting complete illegibility and randomness. In addition, after the deciphering process, the inference process gradually reduces the data volume in the fog sublayer pipeline, as Figure 29a,b evidence. Two detected bounding boxes are shown in Figure 29a; the first shows a person with a pistol, and the second shows a person with a rifle.

7. Conclusions

We conclude that the lightweight, secure GS3 [11] mechanism is a robust and high-performing candidate for the securing of event data transmissions with a short validity period. This research employed GS3 to secure data transmission using SmartFog [17], an efficient distributed system for people surveillance. Based on the current state of knowledge, security communication issues in armed personnel surveillance systems have not been addressed.

The benchmark results demonstrated that GS3 performed image ciphering in a remarkably fast 0.058 s in the edge layer. It was about 38% faster than the AES algorithm. Comparable results were presented for the fog sublayer processing pipeline, where GS3 outperformed AES, reducing the execution time by 70% in the first fog sublayer.

The power consumption measures reaffirm the performance of GS3 for each scenario (P, Pn, AP, APn). For example, GS3 exhibited an overhead of 15%; however, AES introduced a 46% increase in consumption in the Pn scenario.

Although GS3 is slower than Simon or Speck, GS3 has a key space that is much larger. Considering the trade-off between the key space length and speed, GS3 is a much stronger and faster solution than the other methods.

Regarding the security robustness of GS3, it was found that the FCS backwarding and light-signed methods detected 100% of the tampering, replaying, forwarding, and forgery attacks and rejected these blocks. Furthermore, the results in terms of the number of pixels change rate (NPCR) and unified average changing intensity (UACI) tests verified the strength of GS3 against attacks based on differential analysis. In addition, the entropy test and 800-22-rev1 suite test proved the randomness in generating the internal state of the scrambling process.

GS3 maintains full architectural transparency with the underlying armed detection system, ensuring seamless integration without requiring modifications to existing prediction pipelines. Therefore, GS3 is adaptable to different deployment settings.

8. Future Works

Data transfer usually represents the main proportion of the energy consumption in IoT. Thus, data reduction should always be applied whenever possible, especially on edge devices. In future work, we will combine data reduction techniques [73] with the GS3 method. Recently, the field of drones or unmanned aerial vehicles (UAVs) has experienced exponential expansion. Nevertheless, drone technology has associated vulnerabilities and threats, as claimed in [74]. Therefore, we wish to address their security gaps—for example, in the zoom functions of cameras.

Author Contributions: Conceptualisation, J.M.P. and J.O.; methodology, F.A.-V., J.M.P. and J.O.; software, F.A.-V.; validation, F.A.-V., J.M.P., J.O. and F.L.-G.; formal analysis, F.A.-V., J.M.P. and J.O.; investigation, F.A.-V., J.M.P. and J.O.; supervision, J.M.P. and J.O.; writing—original draft preparation, F.A.-V.; writing—review and editing, J.M.P. and J.O.; project administration, J.M.P. and J.O.; resources, J.M.P., J.O. and F.L.-G.; data curation, F.A.-V. and F.L-G. All authors have read and agreed to the published version of the manuscript.

Funding: This work has been partly supported by the Spanish Ministry of Science, Innovation, and Universities, via the "Intelligent distributed processing architectures in Fog level for the IoT paradigm (Smart-Fog)" project, grant RTI2018-098371-B-I00; the ENIA International Chair in Agriculture, University of Córdoba (TSI-100921-2023-3), funded by the Secretary of State for Digitalization and Artificial Intelligence; and the European Union—Next Generation EU, Recovery, Transformation and Resilience Plan.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

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

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

References

1. Taylor, B.G.; Mitchell, K.J.; Turner, H.A.; Sheridan-Johnson, J.; Mumford, E.A. Prevalence of Gun Carrying and Gun Violence Victimization and Perpetration Among a Nationally Representative Sample of U.S. Youth and Young Adults. *AJPM Focus* **2025**, *4*, 100294. [CrossRef] [PubMed]
2. Rezey, M.L.; Olson, D.E.; Stemen, D. Urban Victims of Nonlethal Gun Violence: A Chicago-Centered Analysis Using the National Crime Victimization Survey. *Crime Delinq.* **2025**, *71*, 2389–2416. [CrossRef]
3. Global Terrorism Index. Available online: https://www.visionofhumanity.org/maps/global-terrorism-index/ (accessed on 1 November 2024).
4. Alenizi, A.S.; Al-Karawi, K.A. Internet of Things (IoT) Adoption: Challenges and Barriers. In *Proceedings of the Seventh International Congress on Information and Communication Technology*; Yang, X.S., Sherratt, S., Dey, N., Joshi, A., Eds.; Springer : Singapore, 2023; pp. 217–229.
5. Rejeb, A.; Rejeb, K.; Treiblmaier, H.; Appolloni, A.; Alghamdi, S.; Alhasawi, Y.; Iranmanesh, M. The Internet of Things (IoT) in healthcare: Taking stock and moving forward. *Internet Things* **2023**, *22*, 100721. [CrossRef]
6. Dhanaraju, M.; Chenniappan, P.; Ramalingam, K.; Pazhanivelan, S.; Kaliaperumal, R. Smart Farming: Internet of Things (IoT)-Based Sustainable Agriculture. *Agriculture* **2022**, *12*, 1745. [CrossRef]
7. Firouzi, F.; Farahani, B.; Marinsek, A. The convergence and interplay of edge, fog, and cloud in the AI-driven Internet of Things (IoT). *Inf. Syst.* **2022**, *107*, 101840. [CrossRef]
8. Kong, L.; Tan, J.; Huang, J.; Chen, G.; Wang, S.; Jin, X.; Zeng, P.; Khan, M.; Das, S.K. Edge-computing-driven Internet of Things: A Survey. *ACM Comput. Surv.* **2023**, *55*, 174. [CrossRef]
9. Bukhari, S.M.S.; Zafar, M.H.; Abou Houran, M.; Moosavi, S.K.R.; Mansoor, M.; Muaaz, M.; Sanfilippo, F. Secure and privacy-preserving intrusion detection in wireless sensor networks: Federated learning with SCNN-Bi-LSTM for enhanced reliability. *AD HOC Netw.* **2024**, *155*, 103407. [CrossRef]
10. Alcaraz Velasco, F.; Palomares, J.M.; Olivares, J. Lightweight method of shuffling overlapped data-blocks for data integrity and security in WSNs. *Comput. Netw.* **2021**, *199*, 108470. [CrossRef]
11. Alcaraz-Velasco, F.; Palomares, J.M.; Olivares, J. GS3: A Lightweight Method of Generating Data Blocks With Shuffling, Scrambling, and Substituting Data for Constrained IoT Devices. *IEEE Internet Things J.* **2024**, *11*, 25782–25801. [CrossRef]
12. Piccialli, F.; Di Somma, V.; Giampaolo, F.; Cuomo, S.; Fortino, G. A survey on deep learning in medicine: Why, how and when? *Inf. Fusion* **2021**, *66*, 111–137. [CrossRef]
13. Cui, Y.; Chen, R.; Chu, W.; Chen, L.; Tian, D.; Li, Y.; Cao, D. Deep Learning for Image and Point Cloud Fusion in Autonomous Driving: A Review. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 722–739. [CrossRef]
14. Li, J.; Xu, W.; Deng, L.; Xiao, Y.; Han, Z.; Zheng, H. Deep learning for visual recognition and detection of aquatic animals: A review. *Rev. Aquac.* **2023**, *15*, 409–433. [CrossRef]
15. 5Yadav, P.; Gupta, N.; Sharma, P.K. Robust weapon detection in dark environments using Yolov7-DarkVisionImage. *Digit. Signal Process.* **2024**, *145*, 104342. [CrossRef]
16. Burnayev, Z.R.; Toibazarov, D.O.; Atanov, S.K.; Canbolat, H.; Seitbattalov, Z.Y.; Kassenov, D.D. Weapons Detection System Based on Edge Computing and Computer Vision. *Int. J. Adv. Comput. Sci. Appl.* **2023**, *14*, 812–820. [CrossRef]
17. Martinez, H.; Rodriguez-Lozano, F.J.; León-García, F.; Palomares, J.M.; Olivares, J. Distributed Fog computing system for weapon detection and face recognition. *J. Netw. Comput. Appl.* **2024**, *232*, 104026. [CrossRef]
18. Nechvatal, J.; Barker, E.; Bassham, L.; Burr, W.; Dworkin, M.; Foti, J.; oback, E. Report on the Development of the Advanced Encryption Standard (AES). *J. Res. Natl. Inst. Stand. Technol.* **2001**, *106*, 511–577. [CrossRef]
19. Jamshed, M.A.; Ali, K.; Abbasi, Q.H.; Imran, M.A.; Ur-Rehman, M. Challenges, Applications, and Future of Wireless Sensors in Internet of Things: A Review. *IEEE Sens. J.* **2022**, *22*, 5482–5494. [CrossRef]
20. Singh, S.K.; Kumar Dhurandher, S. Architecture of Fog Computing, Issues and Challenges: A Review. In Proceedings of the 2020 IEEE 17th India Council International Conference (INDICON), New Delhi, India, 10–13 December 2020; pp. 1–6. [CrossRef]
21. Shamshad, S.; Riaz, F.; Riaz, R.; Rizvi, S.S.; Abdulla, S. An Enhanced Architecture to Resolve Public-Key Cryptographic Issues in the Internet of Things IoT, Employing Quantum Computing Supremacy. *Sensors* **2022**, *22*, 8151. [CrossRef]
22. De Cannière, C.; Preneel, B. *New Stream Cipher Designs: The eSTREAM Finalists. (Trivium)*; Springer: Berlin/Heidelberg, Germany, 2008; pp. 244–266.

23. Gundaram, P.K.; Tentu, A.N.; Allu, S.N. State Transition Analysis of GSM Encryption Algorithm A5/1. *J. Commun. Softw. Syst.* **2022**, *18*, 36–41. [CrossRef]
24. Bajaj, N. Linear Feedback Shift Register: 1.0.6. Available online: https://pypi.org/project/pylfsr/ (accessed on 17 April 2023).
25. Wolfram, S. Random sequence generation by cellular automata. *Adv. Appl. Math.* **1986**, *7*, 123–169. [CrossRef]
26. John, A.; Nandu, B.C.; Ajesh, A.; Jose, J. PENTAVIUM: Potent Trivium-Like Stream Cipher Using Higher Radii Cellular Automata. In *Cellular Automata*; Gwizdałła, T.M., Manzoni, L., Sirakoulis, G.C., Bandini, S., Podlaski, K., Eds.; Springer International Publishing: Cham, Switzerland, 2021; pp. 90–100.
27. Dridi, F.; El Assad, S.; El Hadj Youssef, W.; Machhout, M.; Lozi, R. Design, Implementation, and Analysis of a Block Cipher Based on a Secure Chaotic Generator. *Appl. Sci.* **2022**, *12*, 9952. [CrossRef]
28. Alshammari, B.M.; Guesmi, R.; Guesmi, T.; Alsaif, H.; Alzamil, A. Implementing a Symmetric Lightweight Cryptosystem in Highly Constrained IoT Devices by Using a Chaotic S-Box. *Symmetry* **2021**, *13*, 129. [CrossRef]
29. Lorenz, E.N. Deterministic Nonperiodic Flow. *J. Atmos. Sci.* **1963**, *20*, 130–141. [CrossRef]
30. Zhu, S.; Deng, X.; Zhang, W.; Zhu, C. A New One-Dimensional Compound Chaotic System and Its Application in High-Speed Image Encryption. *Appl. Sci.* **2021**, *11*, 11206. [CrossRef]
31. Schneier, B.; Sutherland, P. *Applied Cryptography: Protocols, Algorithms, and Source Code in C*, 2nd ed.; John Wiley & Sons, Inc.: Hoboken, NJ, USA, 1995.
32. Zheng, J. MARC: Modified ARC4. In *Foundations and Practice of Security*; Garcia-Alfaro, J., Cuppens, F., Cuppens-Boulahia, N., Miri, A., Tawbi, N., Eds.; Springer: Berlin/Heidelberg, Germany, 2013; pp. 33–44.
33. The eSTREAM Project, Salsa20. Available online: https://www.ecrypt.eu.org/stream/salsa20pf.html (accessed on 1 April 2024).
34. ChaCha20 and Poly1305 for IETF Protocols. RFC 7539. Available online: hhttps://www.rfc-editor.org/info/rfc7539 (accessed on 5 February 2024).
35. Bogdanov, A.; Knudsen, L.R.; Leander, G.; Paar, C.; Poschmann, A.; Robshaw, M.J.B.; Seurin, Y.; Vikkelsoe, C. PRESENT: An Ultra-Lightweight Block Cipher. In *Cryptographic Hardware and Embedded Systems—CHES*; Paillier, P.,Verbauwhede, I., Eds.; Springer: Berlin/Heidelberg, Germany, 2007; pp. 450–466.
36. Data Encryption Standard DES. Available online: https://csrc.nist.gov/CSRC/media/Publications/fips/46/archive/1977-01-15/documents/NBS.FIPS.46.pdf (accessed on 1 April 2024).
37. Beaulieu, R.; Shors, D.; Smith, J.; Treatman-Clark, S.; Weeks, B.; Wingers, L. The SIMON and SPECK Families of Lightweight Block Ciphers. Cryptology ePrint Archive, Paper 2013/404. Available online: https://eprint.iacr.org/2013/404 (accessed on 5 January 2024).
38. Santos, T.; Oliveira, H.; Cunha, A. Systematic review on weapon detection in surveillance footage through deep learning. *Comput. Sci. Rev.* **2024**, *51*, 100612. [CrossRef]
39. Girshick, R. Fast R-CNN. In Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 11–18 December 2015; pp. 1440–1448. [CrossRef]
40. YOLO. Available online: https://docs.ultralytics.com/models/ (accessed on 2 January 2025).
41. Abdullah, M.; Al-Noori, A.H.Y.; Suad, J.; Tariq, E. A multi-weapon detection using ensembled learning. *J. Intell. Syst.* **2024**, *33*, 20230060. [CrossRef]
42. EfficientDet. Available online: https://github.com/xuannianz/EfficientDet?tab=readme-ov-file (accessed on 12 November 2024).
43. Message Queuing Telemetry Transport. Available online: https://mqtt.org// (accessed on 12 April 2025).
44. Wang, C.Y.; Bochkovskiy, A.; Liao, H.Y.M. YOLOv7: Trainable Bag-of-Freebies Sets New State-of-the-Art for Real-Time Object Detectors. In Proceedings of the 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Vancouver, BC, Canada, 17–24 June 2023; pp. 7464–7475. [CrossRef]
45. You, Y.; Wang, J.; Yu, Z.; Sun, Y.; Peng, Y.; Zhang, S.; Bian, S.; Wang, E.; Wu, W. A Fine-Grained Detection Network Model for Soldier Targets Adopting Attack Action. *IEEE Access* **2024**, *12*, 107445–107458. [CrossRef]
46. Ultralytics YOLOv8. Available online: https://docs.ultralytics.com/es/models/yolov8/ (accessed on 1 March 2024).
47. Amado-Garfias, A.J.; Conant-Pablos, S.E.; Ortiz-Bayliss, J.C.; Terashima-Marin, H. Improving Armed People Detection on Video Surveillance Through Heuristics and Machine Learning Models. *IEEE Access* **2024**, *12*, 111818–111831. [CrossRef]
48. Torregrosa-Dominguez, A.; Alvarez-Garcia, J.A.; Salazar-Gonzalez, J.L.; Soria-Morillo, L.M. Effective Strategies for Enhancing Real-Time Weapons Detection in Industry. *Appl. Sci.* **2024**, *14*, 8198. [CrossRef]
49. Sanjuan, E.B.; Cardiel, I.A.; Cerrada, J.A.; Cerrada, C. Message Queuing Telemetry Transport (MQTT) Security: A Cryptographic Smart Card Approach. *IEEE Access* **2020**, *8*, 115051–115062. [CrossRef]
50. Bouakkaz, F.; Omar, M.; Laib, S.; Guermouz, L.; Tari, A.; Bouabdallah, A. Lightweight Sharing Scheme for Data Integrity Protection in WSN. *Wirel. Pers. Commun.* **2016**, 89, 211–226. [CrossRef]
51. Shannon, C.E. Communication theory of secrecy systems. *Bell Syst. Tech. J.* **1949**, *28*, 656–715. [CrossRef]
52. Farah, T.; Rhouma, R.; Belghith, S. A novel method for designing S-box based on chaotic map and Teaching–Learning-Based Optimization. *Nonlinear Dyn.* **2017**, *88*, 1059–1074. [CrossRef]

53. Artuğer, F.; Özkaynak, F. A method for generation of substitution box based on random selection. *Egypt. Inform. J.* **2022**, *23*, 127–135. [CrossRef]
54. Islam, F.u.; Liu, G. Designing S-Box Based on 4D-4Wing Hyperchaotic System. *3D Res.* **2017**, *8*, 9. [CrossRef]
55. YOLOv5s. Available online: https://github.com/ultralytics/yolov5?tab=readme-ov-file (accessed on 15 January 2025).
56. Wu, C.; Cai, C.; Xiao, F.; Wang, J.; Guo, Y.; Ma, L. YOLO-LSM: A Lightweight UAV Target Detection Algorithm Based on Shallow and Multiscale Information Learning. *Information* **2025**, *16*, 393. [CrossRef]
57. Face-Recognition Lybrary. Available online: https://pypi.org/project/face-recognition/ (accessed on 15 January 2025).
58. Free Videos Shared by the Pexels Community. Available online: https://www.pexels.com/videos/ (accessed on 1 February 2024).
59. Federal Office for Information Security (BSI). *Cryptographic Mechanisms: Recommendations and Key Lengths*, Version: 2023-1; Technical report; German Federal Office for Information Security : Bonn, Germany, 2023.
60. Data Base Images. Available online: https://ccia.ugr.es/cvg/dbimagenes/ (accessed on 2 March 2025).
61. Yudha, U. Source Code of Trivium. Available online: https://github.com/uisyudha/Trivium (accessed on 4 January 2024).
62. Wu, Y.; Noonan, J.P.; Agaian, S.S. NPCR and UACI Randomness Tests for Image Encryption. *Cyber J. J. Sel. Areas Telecommun.* **2011** , *1*, 31–38.
63. Walker, J. A Pseudorandom Number Sequence Test Program. Available online: https://www.fourmilab.ch/random/ (accessed on 4 March 2024).
64. Bassham, L.E.; Rukhin, A.L.; Soto, J.; Nechvatal, J.R.; Smid, M.E.; Barker, E.B.; Leigh, S.D.; Levenson, M.; Vangel, M.; Banks, D.L.; et al. *SP 800-22 Rev. 1a. A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications*; Technical report; National Institute of Standards and Technology (NIST) : Gaithersburg, MD, USA, 2010.
65. Castillo-Secilla, J.M.; Aranda, P.C.; Outeiriño, F.J.B.; Olivares, J. Experimental Procedure for the Characterization and Optimization of the Power Consumption and Reliability in ZigBee Mesh Networks. In *2010 Third International Conference on Advances in Mesh Networks*; IEEE Computer Society: Wahsington, DC, USA, 2010; pp. 13–16.
66. Agilent 3631A Power Suppley. Available online: https://www.keysight.com/es/en/product/E3631A/80w-triple-output-power-supply-6v-5a--25v-1a.html (accessed on 1 March 2024).
67. Vujović, V.; Maksimović, M. Raspberry Pi as a Wireless Sensor node: Performances and constraints. In Proceedings of the 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Opatija, Croatia, 26–30 May 2014; pp. 1013–1018.
68. VISA Protocol. Available online: https://www.ivifoundation.org/specifications/default.html#visa-specifications (accessed on 1 January 2024).
69. IVI Foundation. Available online: https://www.ivifoundation.org/ (accessed on 1 January 2024).
70. PyVISA Library. Available online: https://pyvisa.readthedocs.io/en/latest/introduction/index.html (accessed on 1 January 2024).
71. Pandas Library. Available online: https://pandas.pydata.org (accessed on 1 January 2024).
72. Matplotlib Library. Available online: https://matplotlib.org/ (accessed on 1 January 2024).
73. Leon-Garcia, F.; Palomares, J.M.; Olivares, J. D2R-TED: Data—Domain Reduction Model for Threshold-Based Event Detection in Sensor Networks. *Sensors* **2018**, *18*, 3806. [CrossRef]
74. Mekdad, Y.; Aris, A.; Babun, L.; Fergougui, A.E.; Conti, M.; Lazzeretti, R.; Uluagac, A.S. A survey on security and privacy issues of UAVs. *Comput. Netw.* **2023**, *224*, 109626. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.