

A video stream encryption algorithm that does not propagate errors*

Lyudmila Kovalchuk^{1,†}, Serhii Hilgurt^{1,†} and Hanna Nelasa^{1,*†}

¹ G.E. Pukhov Institute for Modelling in Energy Engineering of the NAS of Ukraine, 15, General Naumov Str. Kyiv, 03164, Ukraine

Abstract

This paper presents a lightweight algorithm intended for temporary masking of UAV video frames in resource-constrained environments. The algorithm performs key-dependent pixel permutations based on modular arithmetic and ensures error non-propagation within the considered pixel-matrix model: corruption of one ciphertext byte affects at most one decrypted pixel. With precomputed lookup tables, encryption and decryption each require one pass over an $m \times n$ frame and have $O(mn)$ computational complexity. For a 360×420 frame, the algorithm-specific auxiliary data occupy approximately 1,568 bytes in a 16-bit implementation. Platform-specific execution time and energy consumption are not measured in this study. Within an explicitly restricted passive ciphertext-only model, the paper estimates the cost of exhaustive search and identifies the loss of security when plaintext-ciphertext coordinate correspondences are available. Security against ciphertext-only statistical attacks remains a subject for further research.

Keywords

video stream encryption, unmanned aerial vehicles, lightweight cryptography, error non-propagation

1. Introduction

Cryptographic protection of a video stream transmitted by a UAV differs substantially from the protection of telemetry, control commands, or individual static images. For video encryption, the critical factors are not only the cryptographic properties of the transformation, but also the end-to-end transmission delay, throughput, amount of RAM, energy consumption, and the behavior of the system in the presence of channel errors. In [1], this problem was formulated for cases in which the operational value of intercepted data is limited by the duration of the UAV mission. In such a setting, a justified trade-off between the security level and speed is admissible, but it must rely on an explicitly defined adversarial model and an estimate of the time required for the corresponding attack.

One conservative variant of multimedia traffic protection is described by Cecchinato et al. [2], who studied the transmission of UAV audio and video data using AES. The authors showed that even for a standard primitive, the result depends on the mode of operation, channel parameters, and the nature of the stream. However, this construction does not aim to increase the speed of the algorithm, for example by reducing the number of rounds with a loss of security; it only provides a baseline, cryptographically conservative variant to which simplified solutions may be compared. In a later study, Cecchinato et al. [3] proposed an architecture for secure multimedia exchange between resource-constrained UAVs using a lightweight implementation of AES and several synchronous keys. This work is important because it addresses real-time operation and adverse radio-channel

* ADT-ISMDDC 2026: *Advanced Digital Technologies: Intelligent Systems, Modeling, and Data-Driven Control*, September 22–24, 2026, Odesa, Ukraine

¹ Corresponding author.

[†] These authors contributed equally.

✉ lusi.kovalchuk@gmail.com (L. Kovalchuk); hilgurt@ukr.net (S.Ya. Hilgurt); annanelasa@gmail.com (H. Nelasa)

ORCID 0000-0003-2874-7950 (L. Kovalchuk); 0000-0003-1647-1790 (S.Ya. Hilgurt); 0000-0002-3708-0089 (H. Nelasa)



© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

conditions. At the same time, its result is architecturally- and implementation-oriented: it does not offer a separate symmetric primitive with temporary security and a speed sufficient for image encryption.

Another implementation-oriented approach is presented by Psilias et al. [4], where an FPGA architecture for protecting UAV video and telemetry based on AES is described. It demonstrates that real-time requirements can be achieved using specialized hardware. However, this does not imply the suitability of such an architecture for a simple software transformation of a frame on inexpensive onboard equipment: its delay, energy consumption, and throughput depend on the specific platform and on the way it is integrated into the transmission path.

The study by Khan et al. [5] is also interesting from a methodological point of view. In that work, stream ciphers for video protection on an ARM processor are compared in terms of frame rate, throughput, cycles per byte, memory usage, and energy consumption. This approach makes it possible to compare the practical suitability of cryptographic mechanisms for protecting video data in resource-constrained systems in a comprehensive way.

The survey by Sarkar et al. [6] confirms that, for UAVs, the choice of a cryptographic mechanism should be based not only on nominal security, but also on delay, energy costs, memory, and deployability in a dynamic environment. In [7], Aljamal et al. proposed a two-layer protection scheme: an adaptive lightweight symmetric mechanism is used for the UAV-gateway channel, while a hybrid ECC-PQC approach is used for the gateway-cloud channel. These publications show the current trend toward context-dependent protection, but they do not provide a concrete solution for encrypting an intensive video stream with low onboard delay.

It should be noted that, for real-time video stream encryption, not only the speed, required memory, and cryptographic security, but also the property of error non-propagation are important. Here, error non-propagation means a property in which the distortion of one bit in the encrypted representation of a pixel after decryption may change the value of only that pixel, and not of other pixels of the frame. This criterion should be distinguished from the cryptographic security of the underlying algorithm. In synchronous stream-cipher mode or counter mode, a single error in the ciphertext does not propagate to subsequent bits of the plaintext; however, localization specifically at the pixel level requires additional assumptions about the format, packing, and synchronization of the frame. In [2-5], the use of AES and stream ciphers for protecting video data is considered, but error localization at the level of a single pixel is not the central subject of these studies.

Related directions represented in [8-10] also do not solve this issue directly: [8] proposes a multicomponent image encryption method, [9] presents an architecture for protecting small drones using public-key mechanisms, and [10] proposes an approach in the white-box model. All of them operate in a different setting and do not establish how a single distortion in encrypted data affects the matrix of the decrypted frame. The properties of SP-networks, in particular the PRESENT cipher [11] and results concerning S-boxes and the diffusion layer [12], are needed for evaluating the security of the underlying primitive, but they do not determine the nature of error propagation in a concrete mode of use.

Thus, the reviewed works do not provide a solution that would simultaneously be simple for software implementation, be lightweight and fast to compute for each frame, allow the justification of temporary security in a specified adversarial model, and ensure localization of a transmission error at the level of a single pixel.

2. Lightweight algorithm for encryption of a rectangular image with temporary security

In this section, we describe a lightweight algorithm with temporary security, intended for the encryption of a rectangular image, and having the following properties.

1. The algorithm does not propagate errors. That is, if distortions occur in the channel during ciphertext transmission and some bytes are transmitted incorrectly, then after decryption the number of incorrect bytes does not increase.
2. The design target is a total encryption–decryption time below 0.3 seconds per frame. This study derives the corresponding computational condition but does not verify it on a specific platform.
3. The amount of auxiliary memory required for implementing this algorithm should not exceed 10^6 bits, excluding the memory needed to store the image itself.
4. The algorithm is intended to provide temporary security: depending on the tasks performed and on the adversary model, the required protection time may range from 2 hours to several days.

The contribution is limited to the mathematical construction and analytical evaluation of a frame-level transformation. Each frame is represented as an $m \times n$ matrix of 8-bit or 24-bit pixel values and permuted directly; codec integration, compression, packetization, and inter-frame coding are outside the scope of this study.

The mathematical basis of the proposed approach is modular arithmetic: the analysis of invertibility of a mapping defined by linear formulas modulo the frame dimensions is reduced to the solvability of systems of linear congruences [13]. The universal modular transformations considered by Yang et al. [14] also show that one-to-one correspondence of a coordinate permutation is determined by the algebraic properties of its parameters. In the proposed algorithm, pixel values are not mixed: only their key-dependent permutation is performed. Therefore, if the pixel boundaries and the data order are preserved, the distortion of one bit in the encrypted representation of a pixel will change one pixel of the decrypted frame at most.

Below, this approach is formalized as a family of key-dependent permutations of pixel positions in a rectangular frame.

2.1. Description of the algorithm

Let $V_l = \{0,1\}^l$; also, for $n \in \mathbb{N}$, denote $[n] = \{0,1,2, \dots, n-1\}$.

We will assume that the screen on which the image is displayed has a rectangular shape and consists of $[m] \times [n]$ pixels. Then the image will be represented as a rectangular matrix Π of dimension $[m] \times [n]$, whose elements are the pixel values π_{ij} :

$$\Pi = (\pi_{ij})_{i,j=1}^{m,n} \quad (1)$$

where, depending on the image type (color or grayscale), $\pi_{ij} \in V_{24}$ or $\pi_{ij} \in V_8$.

In what follows, we identify the image with the matrix (1).

For definiteness, assume that $m < n$, (for example, $m = 360$ and $n = 420$) and denote $d = \gcd(m, n)$.

In our notation, the problem of constructing an image encryption algorithm that permutes pixel values can be formalized as follows. It is necessary to construct a family of transformations $\varphi: [m] \times [n] \rightarrow [m] \times [n]$, where each transformation permutes the pixel values (that is, each of them acts on the set of their indices, $[m] \times [n]$, and is a bijection) and has the following properties:

B1) $\forall i \in [m] \exists 0 \leq j_1 < j_2 \leq n-1: \varphi(i, j_1) = (a_1, b_1)$ and $\varphi(i, j_2) = (a_2, b_2)$, and, moreover, $a_1 \neq a_2$ (that is, the pixel values that were in one row do not necessarily fall into one row again after applying the mapping);

B2) $\forall j \in [n] \exists 0 \leq i_1 < i_2 \leq m-1: \varphi(i_1, j) = (a_1, b_1)$ and $\varphi(i_2, j) = (a_2, b_2)$, and, moreover, $b_1 \neq b_2$ (that is, the pixel values that were in one column do not necessarily fall into one column again after applying the mapping);

B3) this transformation should be specified using a relatively small amount of memory and should be computed quickly.

Each mapping from this family will be specified by a certain set of parameters (the set of such parameter tuples will be the key space). In this case, image encryption for a fixed set of parameters is performed according to the rule:

$$C = E(\Pi) = (c_{i,j})_{i,j=0}^{m-1,n-1}, \text{ where } c_{i,j} = \pi_{\varphi(i,j)} \quad (2)$$

and decryption is performed according to the rule

$$\Pi = D(C) = (\pi_{i,j})_{i,j=0}^{m-1,n-1}, \text{ where } \pi_{i,j} = c_{\varphi^{-1}(i,j)} \quad (3)$$

where φ^{-1} denotes the mapping inverse to φ (it exists because, according to our requirements, the mapping φ is a bijection).

In what follows, we consider a family of mappings, each of which depends on four parameters $p, r, q, s \in [m] \times [m] \times [n] \times [n]$ (this parameter tuple is the key). Thus, for fixed $p, r, q, s \in [m] \times [m] \times [n] \times [n]$, the equality holds:

$$\varphi(i, j) = \varphi_{p,r,q,s}(i, j) = ((i + pj + r) \bmod m, (j + qi + s) \bmod n) \quad (4)$$

Let us consider the encryption algorithm (2), (3), in which the mapping (4) is used. If every tuple of parameters is assumed to be admissible, then the cardinality of the key space $K = [m] \times [m] \times [n] \times [n]$ for this encryption algorithm is $|K| = (m \cdot n)^2$. For small values of m, n , this value may be insufficient even for temporary security: for example, for $m = 360$ and $n = 420$ we obtain $|K| = 360^2 \cdot 420^2 = 22861440000 \approx 1.33 \cdot 2^{34}$, which approximately corresponds to a 34-bit key. If, according to the computational capabilities of the adversary, such security is insufficient, then two or more full or partial iterations of this mapping can be performed (for example, one iteration with four parameters and one with two parameters), each time with different key parameters.

However, we next show that not all parameter tuples are admissible, since the mapping is not bijective for all such tuples.

Let us consider and analyze one iteration.

First, we need to formulate the condition under which the mapping (4) is bijective.

Proposition 1.

The mapping (4) is bijective if and only if the system of congruences

$$\begin{cases} x + py \equiv 0(\text{mod}m); \\ y + qx \equiv 0(\text{mod}n). \end{cases} \quad (5)$$

has the unique solution $(0,0)$ on the intervals

$$-(m-1) \leq x \leq m-1, -(n-1) \leq y \leq n-1.$$

Proof.

First note that the domain and the codomain of the mapping (4) are finite and have the same cardinality. Therefore, bijectivity of this mapping is equivalent to its injectivity.

Assume that for two different pairs, (x_1, y_1) and (x_2, y_2) , the equality $\varphi_{p,r,q,s}(x_1, y_1) = \varphi_{p,r,q,s}(x_2, y_2)$ holds. This means that the following system of equalities is satisfied:

$$(x_1 + py_1 + r) \text{mod} m = (x_2 + py_2 + r) \text{mod} m$$

and

$$(y_1 + qx_1 + s) \text{mod} n = (y_2 + qx_2 + s) \text{mod} n,$$

which is equivalent to

$$\begin{cases} (x_1 + py_1) \text{mod} m = (x_2 + py_2) \text{mod} m; \\ (y_1 + qx_1) \text{mod} n = (y_2 + qx_2) \text{mod} n. \end{cases}$$

Denote $x = x_1 - x_2$ and $y = y_1 - y_2$; then the last system is equivalent to the system

$$\begin{cases} (x + py) \text{mod} m = 0; \\ (y + qx) \text{mod} n = 0. \end{cases}$$

Then the condition that $(x_1, y_1) \neq (x_2, y_2)$ is equivalent to the condition that the system

$$\begin{cases} (x + py) \text{mod} m = 0; \\ (y + qx) \text{mod} n = 0. \end{cases}$$

has a nonzero solution on the intervals

$$-(m-1) \leq x \leq m-1, -(n-1) \leq y \leq n-1,$$

and the proposition is proved.

Note that during a key generation, the condition of Proposition 1 must be checked every time. This condition narrows the key space and, accordingly, reduces its cardinality.

In particular, for the parameters $m = 360$ and $n = 420$, the size of the key space is approximately 2^{32} , which is equivalent to a 32-bit key length (rather than 34 bits, as in the case of choosing parameters without restrictions). Thus, the key space is reduced, but not very substantially.

It is important to note that the bijectivity condition itself does not give a convenient analytical formula for the inverse mapping needed for decryption. However, decryption can be performed without computing the inverse mapping. That is, if encryption is performed according to the rule $c_{i,j} = \pi_{\varphi(i,j)}$, $0 \leq i \leq m-1, 0 \leq j \leq n-1$, (according to (2)), then for decryption it is sufficient for all $0 \leq i \leq m-1, 0 \leq j \leq n-1$, to perform the following steps:

1) compute $\varphi(i, j)$;

2) set $\pi_{\varphi(i, j)} = c_{ij}$.

This decryption method is correct due to the bijectivity of the function φ for each fixed key.

Thus, for correct decryption it is sufficient to verify the bijectivity of the selected mapping; the next step is to optimize its computation.

2.2. Optimization of computations

Since the mapping (4) contains resource-consuming reduction modulo operations (in fact, division with remainder), without optimization such an algorithm is practically impossible to use in a resource-constrained environment. Therefore, we analyze the possibility of increasing the speed of the algorithm by adding an additional amount of memory that will store certain intermediate computation results.

Let, for the mapping $\varphi_{p,r,q,s}(i, j) = ((i + pj + r) \bmod m, (j + qi + s) \bmod n)$, the key data $p, r, q, s \in [m] \times [m] \times [n] \times [n]$ be fixed.

For these fixed key data, let's construct two tables T_p and T_q , each having two rows, whose entries depend on the parameters p and q , respectively.

In the table T_p , the upper row consists of the numbers from 0 to $n - 1$, that is, all values that the variable j can take; the lower row consists of the values $t_p(j) = (p \cdot j) \bmod m$.

In the table T_q , the upper row consists of the numbers from 0 to $m - 1$, that is, all values that the variable i can take; the lower row consists of the values $t_q(i) = (q \cdot i) \bmod n$.

In this notation, the mapping (4) can be rewritten as

$$\varphi_{p,r,q,s}(i, j) = ((i + t_p(j) + r) \bmod m, (j + t_q(i) + s) \bmod n) \quad (6)$$

Using the tables T_p and T_q , the computations on the right-hand side of (6) can be performed as follows, without division modulo:

$$(i + t_p(j) + r) \bmod m = \begin{cases} i + t_p(j) + r, & \text{if } i + t_p(j) + r < m; \\ i + t_p(j) + r - m, & \text{if } m \leq i + t_p(j) + r < 2m; \\ i + t_p(j) + r - 2m, & \text{if } i + t_p(j) + r \geq 2m. \end{cases} \quad (7)$$

Indeed, since each of the values $i, t_p(j), r$, by definition, lies between 0 and $m - 1$, their sum is always strictly less than $3m$; therefore, the computation in the right-hand side of (7) is correct.

Similarly,

$$(j + t_q(i) + s) \bmod n = \begin{cases} j + t_q(i) + s, & \text{if } j + t_q(i) + s < n; \\ j + t_q(i) + s - n, & \text{if } n \leq j + t_q(i) + s < 2n; \\ j + t_q(i) + s - 2n, & \text{if } j + t_q(i) + s \geq 2n. \end{cases} \quad (8)$$

Thus, both coordinates of the image of a pixel can be computed without division modulo, using tables, additions, and at most two subtractions of the corresponding modulus. After that, we can estimate the amount of auxiliary memory required for this acceleration.

2.3. Memory size

To estimate the memory size, consider one frame of size 360×420 pixels, that is, $mn = 151200$ pixels. If the image is stored in grayscale and one byte is allocated per pixel, then one frame requires 151200 bytes, or 1209600 bits. For a color image in RGB format with three bytes per pixel, 453600 bytes, or 3628800 bits, are required. Therefore, if the image buffer itself is taken into an account, the memory size is determined primarily by the frame size, not by the auxiliary data of the algorithm.

The key of one iteration consists of four numbers $p, r \in [m]$ and $q, s \in [n]$. For $m = 360$ and $n = 420$, each of these numbers can be encoded by 9 bits, since $2^8 < 420 < 2^9$. The minimal memory size for the key is 36 bits. In a practical implementation, it is convenient to store these parameters in 16-bit unsigned variables; then the key of one iteration occupies 64 bits, that is, 8 bytes.

For optimized computation of the mapping, two tables are used. The upper rows of the tables do not have to be stored, since they coincide with the array indices; it is sufficient to store only the values $t_{p(j)}, j = 0, \dots, n - 1$ and $t_{q(i)}, i = 0, \dots, m - 1$. Each value is less than $\max(m, n) = 420$; therefore, for theoretical storage, 9 bits are sufficient. Thus, the memory size for the tables is $(m + n) \cdot 9 = 780 \cdot 9 = 7020$ bits, that is, approximately 878 bytes. In an implementation using 16-bit words, the tables occupy $2(m + n) = 1560$ bytes, or 12480 bits.

Thus, the auxiliary memory for one iteration, that is, the key and two tables, even in a 16-bit implementation is approximately 1568 bytes. This is much less than 10^6 bits.

After estimating the auxiliary memory, what remains is to check what order of speed is required to process one frame in real time.

The 1,568-byte estimate covers only algorithm-specific auxiliary data. For a 360×420 frame, separate input and output buffers require 302,400 bytes for grayscale or 907,200 bytes for RGB, yielding total storage of approximately 303,968 or 908,768 bytes, respectively, excluding code, stack, and system overhead. These are analytical requirements, not measured peak RAM values.

2.4. Running time

In the naive implementation, formula (4) contains two reduction modulo operations for each pixel. After the tables T_p and T_q are constructed, these operations are replaced by memory accesses and several additions, comparisons, and subtractions. This is why precomputing the tables is important for a lightweight implementation of the algorithm.

For one frame, $mn = 151200$ pixels have to be processed. One encryption operation performs one pass over all pixels. Decryption can be performed by an analogous pass, computing, for each initial position, its encrypted position and copying the corresponding value from the encrypted image back. Therefore, the total number of positions processed during encryption and decryption of one frame is $2mn = 302400$ pixels.

For the total time of encryption and decryption not to exceed 0.3 seconds, the average processing time of one position must be no more than $0.3/(2mn) \approx 9.9 \cdot 10^{-7}$ seconds, that is, about $1 \mu\text{s}$ per pixel. For a one-byte image this corresponds to processing about 1 Mpixel/s, and for an RGB image to copying approximately 1.8 MB of data in 0.3 seconds, without taking into account the overhead of memory organization and control.

Let C denote the average number of processor cycles required to process one pixel position, including coordinate computation and the corresponding memory transfer, and let f denote the processor clock frequency. The analytical running-time model is then

$$T_{enc} = mnC/f, T_{enc+dec} = 2mnC/f.$$

For $m = 360$ and $n = 420$, the requirement $T_{enc+dec} \leq 0.3$ seconds is equivalent to

$$C \leq 0.3 f / 302400$$

Thus $C \leq 0.992fMHz$ cycles per pixel position: at most about 99, 198, or 397 cycles at 100, 200, or 400 MHz, respectively. These are feasibility thresholds rather than predicted times, because C is platform-dependent.

For a 360×420 grayscale frame, the proposed algorithm performs one coordinate transformation per pixel; seven-round DONATION [15] performs seven 8-bit S -box substitutions per byte plus key addition and diffusion. This is an operation-count comparison, not a throughput ratio, because operation costs are platform-dependent.

Energy satisfies $E = PT_{enc+dec}$, where P is the average active power. Since P and C are platform-dependent, no numerical energy estimate is given without hardware measurements.

3. Security and Model limitations

In this section, we will evaluate the resistance of the proposed algorithm to brute-force attack, as well as discuss potential vectors for other attacks and the limitations of the model in which these evaluations were obtained.

The primary adversarial model considered in this paper is passive and ciphertext-only. The adversary knows the algorithm and the frame dimensions and can intercept encrypted frames. However, during the lifetime of a key, the adversary has no physical access to the UAV, its key storage, or the encryption device, has no access to the corresponding plaintext frames, and cannot submit chosen frames for encryption. Physical capture and key-extraction attacks are outside the scope of this study. In a practical system, their consequences should be mitigated by secure key erasure, short key lifetimes, and key invalidation when the UAV is lost. For completeness, the known-plaintext case is considered below to establish the boundary of the proposed model.

3.1. Security against brute-force attack

Consider a brute-force attack for one iteration under the assumption that the adversary knows the frame size and the general structure of the algorithm, but does not know the parameters p, r, q, s . If the whole parameter space is considered, then for $m = 360$ and $n = 420$, we have $|K| = (mn)^2 = 360^2 \cdot 420^2 = 22861440000 \approx 2^{34.4}$. As noted above, for a strict cryptographic estimate, only admissible keys should be taken into account, for which the corresponding mapping is bijective. Therefore, the value given above should be regarded as an estimate of the full parameter space of one iteration.

If the adversary can test R keys per second, then the approximate time of exhaustive search is $T = |K|/R$. For example, for $R = 10^5$ keys/s we obtain approximately 63.5 hours, for $R = 10^6$ keys/s approximately 6.35 hours, and for $R = 10^7$ keys/s, approximately 38 minutes. The 2-hour threshold corresponds to a speed of about $3.2 \cdot 10^6$ keys/s. Thus, one iteration can provide two-hour security against simple brute force only against an adversary with moderate computational capabilities and without a more efficient way to test the correctness of the key.

If temporary security for several days is required, or if the adversary model allows a much higher key-testing speed, one iteration is insufficient. In this case, it is advisable to apply two or more independent iterations with different key parameters, as well as to limit the lifetime of one key. At the same time, it should be taken into account that for structured transformations, attacks faster than

brute force may be possible; therefore, the estimate $|K|$ alone is not a complete estimate of cryptographic security.

Thus, bruteforce attack provides only the first and simplest level of security evaluation. Next, it is necessary to consider separately the attacks that may use statistical properties of images or the structure of the mapping itself.

3.2. Other attacks

A brute-force attack is not the only relevant attack for such an algorithm. The considered transformation only permutes the pixel values and does not change the values themselves. Therefore, the brightness or color histogram of the image is preserved. For some classes of images, for example frames with large homogeneous regions, a horizon, a road, sky, buildings, or typical contours of objects, even after pixel permutation there may remain statistical features that help the adversary recognize the type of scene or partially recover its structure.

The known-plaintext case defines an important boundary of the proposed security model. Suppose that two plaintext–ciphertext coordinate correspondences are known:

$$c_{i_k, j_k} = \pi_{u_k, v_k}, k = 1, 2.$$

From (4), these correspondences imply

$$u_k - i_k \equiv pj_k + r(\text{mod}m), \quad v_k - j_k \equiv qi_k + s(\text{mod}n)$$

Subtracting the equations for $k = 1$ and $k = 2$ gives

$$p(j_1 - j_2) \equiv (u_1 - i_1) - (u_2 - i_2)(\text{mod}m),$$

$$q(i_1 - i_2) \equiv (v_1 - j_1) - (v_2 - j_2)(\text{mod}n).$$

If

$$\gcd(j_1 - j_2, m) = 1, \quad \gcd(i_1 - i_2, n) = 1$$

these congruences uniquely determine p and q . The remaining parameters are then obtained as

$$r \equiv u_1 - i_1 - pj_1(\text{mod}m), \quad s \equiv v_1 - j_1 - qi_1(\text{mod}n)$$

Thus, two suitably positioned coordinate correspondences are sufficient to recover the key parameters. If the stated coprimality conditions are not satisfied, additional correspondences may be required. Consequently, the proposed algorithm is not secure in the known-plaintext or chosen-plaintext model; its intended applicability is restricted to the passive ciphertext-only model defined above.

At the same time, these limitations do not mean that the algorithm cannot be useful in practical scenarios. Its use can be considered admissible in cases where the UAV is in constant motion and consecutive frames of the video stream change substantially due to changes in camera position, viewing angle, object scale, lighting, or background. In this regime, the adversary does not obtain a long sequence of almost identical images from which the structure of the permutation could gradually be recovered or stable scene contours could be used. The admissibility of this approach is additionally increased if the key parameters are changed frequently, for example after a fixed number of frames or after short time intervals.

Thus, this algorithm should be considered as a means of temporary masking of a video stream under limited computational resources, rather than as a universal algorithm for cryptographic protection of images. Its most natural area of application is scenarios in which the adversary has

only passive access to the transmitted video stream, has no ability to obtain pairs of open and encrypted frames for the same key, and the usefulness of the intercepted information decreases rapidly over time. In contrast, for static images, repeated scenes, video from a fixed camera, or prolonged use of the same key, this algorithm may be insufficient, since pixel permutation does not completely destroy the statistical and geometric structure of the image.

These considerations define the limits of applicability of the algorithm and naturally lead to the formulation of the model limitations.

3.3. Model limitations

This study provides only the mathematical construction and analytical operation-count and storage estimates. No prototype on target UAV hardware was available; therefore, execution time, peak RAM, power, and energy were not measured. Experimental evaluation on a specified platform is left for future work.

The estimates of running time and memory given above are preliminary. They are based on counting elementary operations and the amount of auxiliary data, but they do not take into account all features of the real UAV hardware platform: processor frequency, cache memory, memory bandwidth, frame format, method of access to the video buffer, energy consumption, and interaction with other processes. Therefore, a final conclusion about the practical applicability of the algorithm requires experiments on real hardware platforms or on platforms as close as possible to real ones.

The error non-propagation property also has clear limits of applicability. It holds for the model in which the encrypted image is transmitted as a sequence of independent bytes or pixel components, and corruption of an individual byte does not change the syntax of the entire stream. If, before or after transmission, compression, packetization, coding with inter-frame dependencies, or another format is used in which an error in one bit can affect the decoding of a larger fragment, then the error non-propagation property must be verified for the entire transmission system, not only for the pixel permutation.

In addition, the described transformation by itself does not ensure authenticity or integrity of the data. If the adversary intentionally changes, deletes, or permutes bytes of the encrypted stream, the algorithm can localize the consequences of individual distortions, but it does not detect the fact of active interference. Protection against an active adversary requires additional mechanisms for integrity control and authentication.

Thus, the proposed approach should be regarded as a lightweight algorithm for temporary protection or masking of video data within a clearly defined model: short key lifetime, limited computational capabilities of the adversary, no access to chosen open images, and transmission of frames in a format compatible with the error non-propagation requirement. The final evaluation of the viability of the algorithm should include not only theoretical calculations, but also experimental measurements of time, memory, energy consumption, and the quality of visual masking on real video streams.

4. Conclusions

This paper proposes a lightweight algorithm for temporary protection of video frames based on a key-dependent permutation of pixel values in a rectangular image. For the considered family of transformations, a bijectivity condition is formulated that ensures correct decryption. Encryption and decryption can be performed without explicitly computing the inverse mapping, and both have computational complexity $O(mn)$. For a 360×420 grayscale frame, the proposed algorithm performs one coordinate transformation per pixel; seven-round DONATION [15] performs seven S-box

substitutions per byte plus key addition and diffusion. This analytical operation-count comparison is not a measured throughput ratio. However, this operation-count ratio should not be interpreted as a measured sevenfold throughput advantage, since the actual speedup depends on the target platform, memory organization, and implementation details. The proposed construction also does not propagate transmission errors: distortion of an individual ciphertext byte does not increase the number of distorted bytes after decryption.

The security analysis establishes important limitations of the proposed transformation. Exhaustive search over the admissible key parameters provides only a baseline estimate and does not exclude attacks that exploit the algebraic structure of the mapping. In particular, the direct parameter-recovery analysis shows that two suitably positioned plaintext–ciphertext coordinate correspondences are sufficient to recover the parameters p, q, r and s when the corresponding coordinate differences are invertible modulo m and n . If these invertibility conditions are not satisfied, additional correspondences may be required. Consequently, the proposed algorithm is not secure against known-plaintext or chosen-plaintext attacks.

Therefore, the possible applicability of the algorithm is restricted to the passive ciphertext-only model specified in this paper. In this model, the adversary can intercept encrypted frames and knows the algorithm and frame dimensions, but has no access to the corresponding plaintext frames, cannot submit chosen frames for encryption, and has no physical access to the UAV, its key storage, or the encryption device during the lifetime of the key.

Security in this restricted model has not yet been fully established. Since the transformation permutes pixel positions without changing pixel values, ciphertext-only statistical attacks may potentially exploit the redundancy of individual images and temporal correlations between consecutive video frames. Determining whether such attacks can recover useful information, estimating their data and computational complexity, and establishing appropriate key lifetimes remain subjects for further research.

The study gives the exact auxiliary-memory requirement and a parametric condition for the target processing time; platform-specific execution-time and energy measurements are left for future work. The comparison with DONATION presented in this paper is based on the number of principal data-transformation operations and does not constitute an experimental throughput or energy-consumption comparison. A practical evaluation should therefore include measurements of execution time, memory footprint, and energy consumption on a specified target platform and for representative grayscale and RGB video frames. Finally, the error non-propagation property characterizes only the behavior of the transformation in the presence of transmission errors and should not be interpreted as a substitute for cryptographic security analysis.

Acknowledgements

The authors gratefully acknowledge the financial support provided by the National Research Foundation of Ukraine, under which this study was carried out within the framework of Project No. 2025.07/0204, "Secure jamming-resistant system for transmitting video information from an unmanned aerial vehicle," state registration No. 0125U003164.

Declaration on Generative AI

During the preparation of this work, the authors used GEMINI in order to: Grammar and spelling check. After using these tool, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

- [1] L. Kovalchuk, A. Davydenko, T. Klymenko, A. Nedashkivska, S. Hilgurt, Development of an Algorithm with Temporary Cryptographic Security for Encrypting Video Stream from an Unmanned Aerial Vehicle, *Eastern-European Journal of Enterprise Technologies* 5 (2025) 41–53. doi:10.15587/1729-4061.2025.340917.
- [2] N. Cecchinato, A. Toma, C. Drioli, G. Oliva, G. Sechi, G.L. Foresti, A Secure Real-Time Multimedia Streaming through Robust and Lightweight AES Encryption in UAV Networks for Operational Scenarios in Military Domain, *Procedia Comput. Sci.* 205 (2022) 50–57. doi:10.1016/j.procs.2022.09.006.
- [3] N. Cecchinato, A. Toma, C. Drioli, G. Oliva, G. Sechi, G.L. Foresti, Secure Real-Time Multimedia Data Transmission from Low-Cost UAVs with A Lightweight AES Encryption, *IEEE Commun. Mag.* 61 (2023) 160–165. doi:10.1109/MCOM.001.2200611.
- [4] D. Psilias, A. Milidonis, G. Lentaris, I. Voyiatzis, Secure Video and Telemetry FPGA Architecture for UAVs, *Proceedings of the 2024 9th South-East Europe Design Automation, Computer Engineering, Computer Networks and Social Media Conference (SEEDA-CECNSM)*, Preveza, Greece (2024) 69–74. doi:10.1109/SEEDA-CECNSM63478.2024.00021.
- [5] M. Khan, H. Dagenborg, D. Johansen, Performance Evaluation of Lightweight Stream Ciphers for Real-Time Video Feed Encryption on ARM Processor, *Future Internet* 16 (2024) 261. doi:10.3390/fi16080261.
- [6] S. Sarkar, S. Shafaei, T.S. Jones, M.W. Totaro, Secure Communication in Drone Networks: A Comprehensive Survey of Lightweight Encryption and Key Management Techniques, *Drones* 9 (2025) 583. doi:10.3390/drones9080583.
- [7] M. Aljamal, B.S. Khassawneh, A. Alsarhan, S. Okour, L.A. Almusfar, B.F. AlThani, W. Aldossary, A Novel Dual-Layer Quantum-Resilient Encryption Strategy for UAV-Cloud Communication Using Adaptive Lightweight Ciphers and Hybrid ECC-PQC, *Computers* 15 (2026) 101. doi:10.3390/computers15020101.
- [8] A. Belazi, H. Migallón, Drone-Captured Wildlife Data Encryption: A Hybrid 1D-2D Memory Cellular Automata Scheme with Chaotic Mapping and SHA-256, *Mathematics* 12 (2024) 3602. doi:10.3390/math12223602.
- [9] M.O. Ozmen, E. Yavuz, Dronecrypt - An Efficient Cryptographic Framework for Small Aerial Drones, *Proceedings of the 2018 IEEE Military Communications Conference (MILCOM)*, Los Angeles, CA, USA (2018) 1–6. doi:10.1109/MILCOM.2018.8599784.
- [10] Y. Yang, H. Dong, Z. Li, S. Xiao, A Lightweight White-Box Encryption Device with CARX Algorithm for Secure Image Transmission, *J. King Saud Univ. Comput. Inf. Sci.* 35 (2023) 101727. doi:10.1016/j.jksuci.2023.101727.
- [11] A. Bogdanov, L.R. Knudsen, G. Leander, C. Paar, A. Poschmann, M.J.B. Robshaw, Y. Seurin, C. Vikkelsøe, PRESENT: An Ultra-Lightweight Block Cipher, in: *Cryptographic Hardware and Embedded Systems – CHES 2007, Lecture Notes in Computer Science* 4727, Springer, Berlin/Heidelberg, Germany (2007) 450–466. doi:10.1007/978-3-540-74735-2_31.
- [12] H.M. Heys, S.E. Tavares, Substitution-Permutation Networks Resistant to Differential and Linear Cryptanalysis, *J. Cryptol.* 9 (1996) 1–19. doi:10.1007/BF02254789.

- [13] N. Koblitz, A Course in Number Theory and Cryptography, 2nd ed., Graduate Texts in Mathematics, Springer, New York, NY, USA (1994). doi:10.1007/978-1-4419-8592-7.
- [14] X. Yang, X. Yu, Q. Zou, J. Jia, Image Encryption Algorithm Based on Universal Modular Transformation, Information Technology Journal 9 (2010) 680–685. doi:10.3923/itj.2010.680.685.
- [15] L. Kovalchuk, T. Klymenko, A. Davydenko, S. Hilgurt, Lightweight Block Cipher DONATION: Evaluation Of Performance And Resource Consumption, in: The Fourth International Scientific and Practical Conference “Cyberwarfare: Intelligence, Defense and Offensive Security” (2026) 34–35.