Journal of Computer Science

Original Research Paper

New Algorithm for Digital Video Encryption

[1]Samsul Arifin, [2]Wihikanwijna, [3]Tuga Mauritsus, [4]Suwarno, [5]Felix Indra Kurniadi, [5]Muhammad Amien Ibrahim, [6]Indra Bayu Muktyas and [6]Nerru Pranuta Murnaka

[1]*Department of Statistics, School of Computer Science, Bina Nusantara University, Jakarta, 11480, Indonesia*
[2]*GONG Yogyakarta, Jalan Kp. Sitisewu, Yogyakarta, 55272, Indonesia*
[3]*Master of Information Systems Management, Bina Nusantara University, Jakarta, 11480, Indonesia*
[4]*Department of Primary Teacher Education, Faculty of Humanities, Bina Nusantara University, Jakarta, Indonesia*
[5]*Department of Computer Science, School of Computer Science, Bina Nusantara University, Jakarta, 11480, Indonesia*
[6]*Department of Mathematics Education, STKIP Surya, Tangerang, 15115, Indonesia*

Article history
Received: 07-03-2023
Revised: 18-04-2023
Accepted: 29-05-2023

Corresponding Author:
Samsul Arifin
Department of Statistics, School of Computer Science, Bina Nusantara University, Jakarta, 11480, Indonesia
Email: samsul.arifin@binus.edu

Abstract: This research aims to propose a safe digital video data alternative employing symmetric cryptography and an encryption technique using Partition. The Hill Cipher method requires a square key matrix with an inverse modulo the unimodular matrix is one of the unique matrices with an inverse. The encryption mechanism is modulo matrix multiplication, with shift cipher encryption employed to encrypt defective partitions. The results reveal that the videos are well encrypted and difficult for third parties to read the partition encryption technology assures that the encrypted and decrypted file sizes are the same. The background of this topic is that digital data security is increasingly important due to the increasing use of digital technology. This research aims to develop a more secure and effective video encryption algorithm by combining several cryptographic methods such as Hill cipher, partition, the shift cipher, unimodular matrix, and binary file concept logistic function. The method used in this study is experimental by using the Python programming language to implement the encryption algorithm. The trials were carried out by comparing the performance of the algorithms developed using different key sizes and variations of the combination methods used. The results show that the developed algorithm can provide a high level of security in the video encryption process with good effectiveness. The use of a combination of different cryptographic methods also has a positive impact on the resulting level of security. Therefore, the developed encryption algorithm can be a good alternative for use in securing sensitive video data.

Keywords: Python, Custom Logistic Map, Hill Cipher, Unimodular Matrix, Partition

Introduction

Digital videos are quite significant in this decade. While sharing digital videos from one person to another, the process of security is typically conducted. In low-security broadcasts, we'll make an effort to draw attention to the level of security. This necessitates greater security for sending and storing digital videos. Digital video encryption is one approach. The Hill Cipher is a well-known encryption algorithm. The Hill Cipher has lately undergone a number of changes. They are the Hill Cipher in combination with a genetic algorithm, the Hill Cipher in combination with image block randomization, and pixel value transformation the Hill Cipher in combination with a chaotic function. In this strategy, the sole finite key matrix employed is either 3×3-4×4. Finding the inverse key or invertible key matrix is said to be difficult or time-consuming if the key matrix size exceeds four (Jarjar *et al*., 2020; Obaida *et al*., 2022).

A special matrix, known as a unimodular matrix, can be used to resolve this problem. We employ Basic Row Operations to produce a unimodular matrix. It's not necessary to use the complete matrix as a key. We will create a unimodular matrix using a Custom Logistic Map because fewer components are required. The Python 3.10 programming language will also be used to implement the suggested method on a number of standard grayscale and color pictures. Learning Python is a rather simple process. This application is also quite easy to get. Only a few operating systems, including Windows, Linux, Mac OS Android, support Python. Python has several applications across a wide range of fields of study and skill sets. These

Science Publications

© 2023 Samsul Arifin, Wihikanwijna, Tuga Mauritsus, Suwarno, Felix Indra Kurniadi, Muhammad Amien Ibrahim, Indra Bayu Muktyas and Nerru Pranuta Murnaka. This open-access article is distributed under a Creative Commons Attribution (CC-BY) 4.0 license.

are some of the justifications for using Python (Muktyas and Arifin, 2018; Arifin *et al*., 2022a). We discovered a flaw in an earlier study, specifically the constrained key space for password 1. This is so because the size of the digital video is a factor that is related to password1. The only sizes that are possible are 1, p_1, p_2 and p_1p_2 if the video size is p_1p_2, where p_1 and p_2 are prime values. By combining the Shift 128 cipher and the Hill Cipher, this can be avoided (Muktyas *et al*., 2021).

There are a few things that need to be explained in relation to the ensuing questions. When unimodular matrix encryption over Z_{256} is just as effective, why do we need to apply shift cipher 128 encryption and combination encryption techniques? The response is given below. Partitions and encryption techniques are used first. Encryption techniques with Partitions allow us to perform the encryption process quickly and efficiently without overburdening the performance of the machine (computer). With Partition Encryption Technique, we encrypt files in small parts. This is more efficient than if we encrypt files with (possibly) very large sizes. The analogy to the way we eat large meals. Of course, we will eat it in small pieces because our mouths will find it difficult to chew large foods efficiently. Second, shift cipher 128 encryption is required because partitioning a file could lead to an improper partition. Shift cipher 128 is used to encrypt the problematic partition. Especially if the imperfect partition size's prime is relatively large (Rihartanto *et al*., 2020; Elkamchouchi *et al*., 2020).

The research domain of this topic is cryptography and information security, particularly in encryption techniques to protect digital video from unauthorized access (Paragas *et al*., 2019). In addition, this topic can also be included in the multimedia field, especially in the video encryption process. The novelty of this research is the use of a combination of several pre-existing cryptographic methods, namely Hill cipher, partition, shift cipher, and unimodular matrix logistic functions in the video encryption process. The combination of several cryptographic methods can increase security and encryption resistance against attacks from irresponsible parties (Rajvir *et al*., 2020).

The background of this topic is that digital data security is increasingly important along with the increasing use of digital technology. Digital video is becoming an increasingly popular form of digital data and needs to be protected from unauthorized access. Therefore, it is important to develop stronger and more efficient encryption algorithms to protect digital videos from attacks. Hill cipher method, partition, shift cipher, and unimodular matrix logistic function are some of the encryption techniques that have been developed before and have proven to be strong enough to protect digital data (Yang *et al*., 2020). However, the combination of these techniques can significantly improve encryption security. In addition, previous research on video encryption has yielded several techniques, but most of them have not been fully effective or practical in real applications. Therefore, there is a need to continue to develop better encryption techniques to protect digital videos in a more effective and efficient way. In this case, the combined implementation of the Hill cipher method, partition, shift cipher, and unimodular matrix logistical functions in the video encryption process can be an important contribution to the development of stronger and more efficient video encryption techniques (Ibrahim *et al*., 2021).

Some of the reasons for the importance of the work proposed in this topic include the following. Cryptography is a very important field for maintaining the confidentiality of data and information, especially in the increasingly advanced digital era. The use of a combination of several strong cryptographic methods can provide better protection for digital data, especially for sensitive or confidential data. Video is an increasingly used and important form of digital data, especially in the multimedia and entertainment fields (Suresh and Ratheesh, 2020). Therefore, protecting videos is becoming increasingly important, especially in terms of security and privacy. The combination of several strong cryptographic methods such as Hill cipher, partition, shift cipher, and unimodular matrix logistic functions can provide a higher level of security for video encryption because each method has different advantages and disadvantages. This research can contribute to the development of stronger and more effective cryptographic methods to protect digital data, especially in terms of using complex combinations of cryptographic methods. The results of this study can be useful for application and system developers who require a high level of security for digital data, especially in the multimedia and entertainment fields (Dooley, 2018).

In this study, a method for encrypting video is proposed that combines the Hill cipher, partition, shift cipher, and unimodular matrix logistic functions (Rahman *et al*., 2013). Some of the quantitative advantages of this method include:

(a) Data security: By using this combination method, the level of data security in the video encryption process can increase, because the combination of several different cryptographic methods will provide stronger security than using a single method alone. (b) Better performance: Implementation of this combination makes it possible to perform the encryption process with better performance and faster because the use of several different cryptographic methods can speed up the encryption process. (c) More efficient use of resources: This method can help use resources more efficiently, such as memory and CPU usage, so that the video encryption process can run more smoothly and does not take up much time and resources. (d) Compatibility with future technologies: This method can also be adapted to future technological developments, so as to provide better security and performance in future application development (Basavaiah *et al*., 2021).

In the context of the work proposed in this study, there are several methods/approaches used, namely: (a) Hill Cipher: This method is used to encrypt data using a key matrix and a plaintext matrix. This method is one of the classic cipher methods which is fairly safe. (b) Partition: This method is used to divide the data to be encrypted into several parts of the same size. This is done to simplify the encryption process. (c) Shift Cipher: This method is used to shift the characters in the text by a certain number of positions to produce encrypted text. (d) Unimodular Matrix: This method is used to generate a secure key matrix that is not easily guessed by unauthorized parties. (e) Logistics Function: This method is used as a scrambling algorithm in the data encryption process. The logistic function can generate random numbers with an unpredictable distribution. By combining these methods, it is expected to produce an encryption system that is more secure and effective in securing data on videos (Muktyas *et al*., 2021; Arifin *et al*., 2021).

In this research, we propose a Python-based encryption technique for digital films based on unimodular matrices and logistic maps (Delmi *et al*., 2020). The topic of research methodologies is covered first, followed by an analysis of the theory applied and our Python code. The discussion of the algorithm's implementation is followed by some analysis the paper ends with a Conclusion session (Arifin *et al*., 2022b). The specific contributions of this study are as follows. (a) Developing a new method for securing video data by using a combination of Hill cipher, partition, shift cipher, and unimodular matrix logistic functions methods. (b) Improves video data security with more complex encryption methods. (c) Provides a new alternative to existing video encryption methods. (d) Proving the effectiveness and superiority of the proposed method in securing video data (Muktyas *et al*., 2021).

This study suggests that implementing the Hill cipher, partition, shift cipher, and unimodular matrix logistic functions in tandem with video encryption can increase data security on the video and prevent unwanted access to it. The limitations of this study, the test was only carried out on videos with certain formats and had not been tested on different video formats. In addition, the test was only conducted at certain video sizes and has not been tested for larger video sizes (Hussein and Amintoosi, 2023).

In this study, we use the flow as follows. The introductory session contains the background, problem formulation, aims, and benefits of this research. In the Methodology session, we discussed the concept of Hill Cipher, partition, shift cipher, and unimodular matrix logistic function. Furthermore, we also discuss the implementation of the program code that we created during the digital video encryption and decryption process. In the results and discussion section, we examine the results obtained in this research and discuss them. Write this ending with conclusions, suggestions open issues contained in the Conclusion section (Arifin and Muktyas, 2018).

Materials and Methods

Several methods/approaches that can be used in this research are (a) Hill cipher method for text encryption. (b) Partition method to break the video into small blocks. (c) Shift cipher method to shift characters in the text. (d) Unimodular matrix to ensure proper encryption and decryption. (e) Logistics function to generate random keys in the encryption and decryption process. Some of the inspiration that can be drawn from the work on this topic includes A combination of several cryptographic methods to increase the security of data encryption. Application of the concept of unimodular matrices to cryptography to generate random keys. The use of logistical functions in the data encryption process increases the complexity of the encryption process and the difficulty of decryption. The use of cryptographic technology in video is a form of developing the use of cryptographic technology in the multimedia field (Muktyas *et al*., 2021; Arifin *et al*., 2021).

Plaintext was encrypted by Lester Hill using a system of linear equations. The Hill cipher divides plaintext into a number of blocks prior to encrypting it. The SLE with n equations and n variables modulo m, where m is an integer, is solved to produce ciphertext when given an element plaintext block. Matrix multiplication could be used to finish the SLE. Due to the symmetrical cryptography used by the Hill Cipher, the created key must have an inverse (Hanson, 1982). The following Fig. 1 is a view of the folder containing the application we are developing.

In this study, the machine's terminology and Python's terms are as follows. Machine specifications must be used by a laptop or computer. Using a machine with 8 GB RAM and Ryzen 3100 processor, this application was created. System requirements for operating the Windows operating system must be installed on the PC or laptop. The operating system Windows 10 Pro 21H1 was used to create this application. Python Prerequisites Python 3 must be installed on the computer or laptop being used (Arifin and Garminia, 2019). Make that the NumPy and tqdm packages for Python have been installed if Python 3 has been installed. For further information, see the figure. If the package hasn't already been installed, do so using the steps below. Please launch the Command Prompt to install the NumPy package. After entering pip install NumPy, press the Enter key. need to set up the tqdm package. Launch the command prompt. Once you've typed pip install tqdm, hit the enter key (Oliphant, 2007).

Following that, we'll discuss a unique matrix known as a unimodular matrix, which will act as the study's key matrix. We'll look at unimodular matrices and how to create them.

DOI: 10.3844/jcssp.2023.847.860

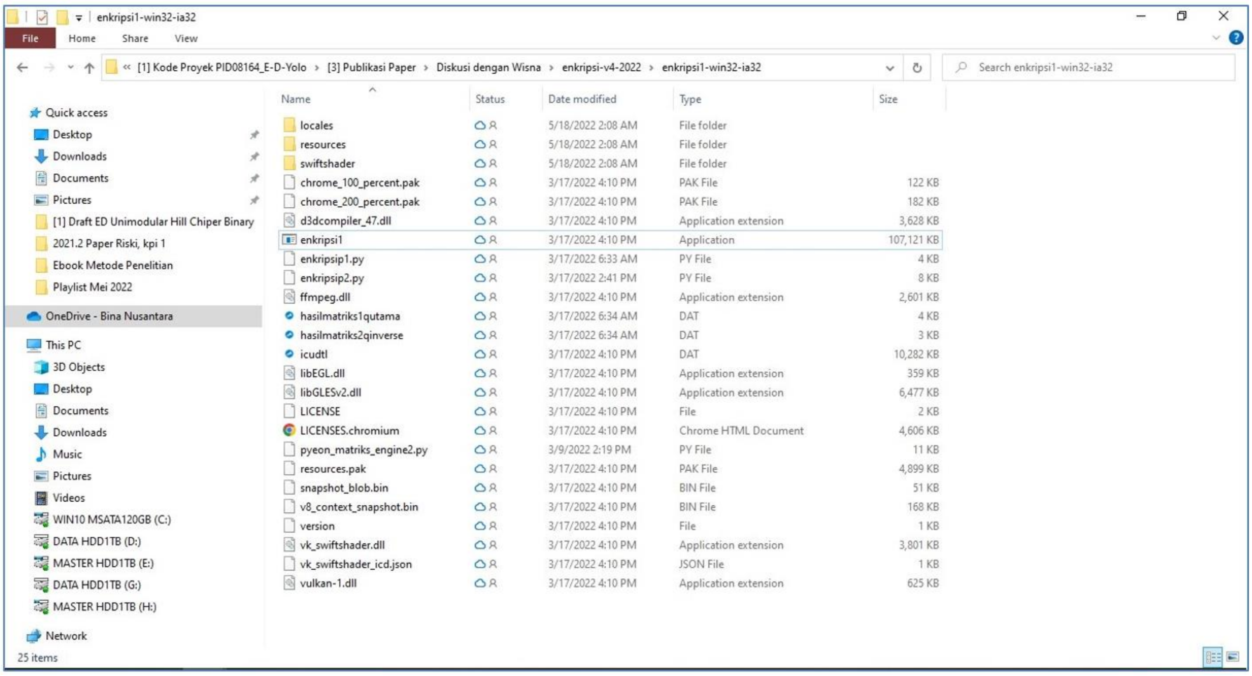

Fig. 1: The folder containing the application we are developing

According to Harrison 1982, if $det(A) = -1$ or $det(A) = 1$, a matrix A with integer entries for each element is considered to be unimodular. Unimodular matrices include the identity matrix, upper triangular matrix lower triangular matrix, with diagonal entries of 1 or -1. The following theorem, which supports this, reads (Arifin and Muktyas, 2018):

$$if\ A_{n\times n}\ is\, a\, triangular\ matrix\, then\, it\ applies\, det(A) = a_{11}.a_{11}.....a_{nn}$$

Following are the steps for constructing a unimodular matrix of size $n\times n$ using Python: (a) Make a diagonal matrix using the entries in the diagonal $a_{ii} = 1$ or $a_{ii} = -1$. (b) For each element in a_{ij}, enter any integer with $i < j$. As a result, an upper triangular matrix with a determinant of 1 or -1 has been created. This matrix has only one module. (c) In order for a matrix to be complete, employ simple row operations or simple column operations going from the final row or column to the first row or column (Komosko *et al*., 2016).

The encryption method with partitions that will be employed in this study will next be covered. Please note that a file is composed of bytes. The value of one byte is an integer from 0-255. Partitioning is dividing a file into smaller parts. The small part is called a partition. For example, a simple example is as follows. We have a file size of 418 bytes. If we want to partition the file with a partition that is 128 bytes long, we will get 4 partitions. The acquisition of the number 4 is explained as follows. Yes, the trick is to divide 418 by 128. However, 418 is not divisible by 128. Moreover, 418 divided by 128 is 3.265625. Yes, because the result of the division (3.265625) is rounded up. 3.265625 is rounded up to 4. The four partitions are 1^{st} partition: 1^{st} bytes to 128^{th} bytes. 2^{nd} partition: 129^{th} bytes to 256^{th} bytes. 3^{rd} partition: 257 bytes to 384 bytes. 4^{th} partition: 385 bytes to 418 bytes. Note that the 1^{st}, 2^{nd} and 3^{rd} partitions are the same length, which is 128 bytes. While the 4^{th} partition has a length of 34 bytes. The 4^{th} partition is referred to as an imperfect partition. The 1^{st}, 2^{nd} and 3^{rd} partitions are called perfect partitions (Obaida *et al*., 2022).

Partitions make the encryption easier for us to encrypt a file. We will partition a source file in such a way that it will only result in at most one imperfect partition. The method is as follows. We first set the partition size, which is q bytes. If our source file is N bytes, then: (1) If N is divisible by q, then we will have partitions of N/q which are all perfect partitions (2) If N is not divisible by q, then we will have partitions several $(N-(N \bmod q))/q$ which are perfect partitions and 1 imperfect partition. Using the unimodular matrix encryption technique on the Z_{256}, we encrypt all perfect partitions one by one. 1^{st} perfect partition encrypted 1^{st} perfect partition 2^{nd} perfect partition encrypted 2^{nd} perfect partition... and so on. If there is an imperfect partition, then we can encrypt the imperfect partition using the shift cipher-128 encryption technique. Incomplete partition encryption result. The encryption result is a combination of the results of the previous two steps. Examples of encryption techniques with partitions are as follows. Suppose we have a file size of 18 bytes to be encrypted. The file structure is as follows: 1st byte = 133, 2nd byte = 57, 3rd byte = 91, 4th byte =19, 5th byte = 0, 6th byte = 211, 7th byte = 70, 8th byte = 11, 9th byte = 104, 10th byte = 67, 11th byte = 78, 12th byte = 86, 13th byte = 112, 14th byte = 51, 15th byte = 0, 16th byte = 133, 17th byte = 11 18th byte = 90 (Jameel and Fadhel, 2022).

The steps are as follows: Step 1. We set the partition size to be 4 bytes. Thus we will have 4 perfect partitions and 1 imperfect partition. (a) The 1st perfect partition contains the 1st-4th bytes: 133, 57, 91, 19 (b) 2nd perfect partition contains 5th-8th bytes: 0, 211, 70, 11 (c) 3rd perfect partition contains 9th-12th bytes: 104, 67, 78, 86 (d) The 4th perfect partition contains the 13th-16th bytes: 112, 51, 0, 133 and (e) The imperfect partition contains 17th-18th byte: 11, 90. Step 2 is up next. We encrypt all 1st-4th Perfect Partitions using the unimodular matrix encryption algorithm over Z_{256}. As an example: (a) The first perfect partition encryption with a unimodular matrix encryption algorithm over Z_{256} yielded 54, 14, 90, 211. (b) The results of the second perfect partition encryption with unimodular matrix encryption over Z_{256} are 244, 142, 16, 25 244. (c) The third perfect partition encryption with unimodular matrix encryption approach over Z_{256} yielded 66, 67, 114, 115 (d) The results of the fourth perfect partition encryption using unimodular matrix encryption over Z_{256} are 91, 92, 93 94. Step 3 is as follows. Because there is an imperfect partition, we can encrypt the imperfect partition using the shift cipher-128 encryption technique. For example, the result of imperfect partition encryption with shift cipher-128 encryption technique is 43, 143. Finally, Step 4. The result of the encryption is a combination of the results in step 2 and 3. Here is the arrangement of the bytes of the encrypted file using the partitioning technique (a) 1st byte = 54 6th bytes = 142 11th bytes = 114 16th bytes = 94, (b) 2nd byte = 14th bytes 7th = 16th bytes = 115th bytes 17th = 43, (c) 3rd byte = 90 8th bytes = 13th 25 bytes = 91 18th bytes = 143, (d) 4th byte = 211 9th byte = 66 14th bytes = 92 (d) 5th byte = 244 10th bytes = 67 15th bytes = 93 (Jameel and Fadhel, 2022).

Please pay some attention to the color of the numbers above. The i-th byte in the j-th partition in the source file will correspond to the i-th byte of the j-th partition in the encrypted file. Then we'll go over the shift cipher 128 encryption method. Let's go through some fundamental algebraic structural concepts. Keep in mind that $Z_{256} = \{0.1, 2, 3, 4, 255\}$. Z_{256} is the group for addition operations modulo 256. The modulo 256 multiplication operation is not opposed by the group Z_{256}. Plain text is text that has not been encrypted. After the text has been encrypted, it is known as ciphertext (Anton, 2018).

This is the shift cipher 128 encryption method. If the raw text has 256 elements, shift cipher 128 will encrypt it by multiplying each element by 128 (modulo 256). Consider the case below. 5 character known plain text string: 213, 110, 7, 91, 65. The plain text will be encrypted using shift cipher 128. The procedures are as follows. In plain text, the first character is 213. Add (modulo 256) 213-128 to get (341) mod 256 = 85. The plain text encryption result for the first character is 85. 85 is the ciphertext's character -1. In plain text, the second character is 110. Add (modulo 256) 110-128 to get (238) mod 256 = 238. In plain text, the encryption result for the second character is 238. The character -2 in the ciphertext is 238. In simple text, the third character is 7. 7 +mod 256 128 = (135) mod 256 = 135. 7 +mod 256 128 = (135) mod 256 = 135. In plain text, the encryption result for the third character is 135. In the ciphertext, 135 is the third character. In plain text, the fourth character is 91. Add (modulo 256) 91-128 to get (219) mod 256 = 219. In plain text, the encryption result for the fourth character is 219. The character -4 in the ciphertext is 219. In plain text, the fifth character is 65. 65 +mod 256 128 = (193) mod 256 = (193) mod 256 = 193. In plain text, the encryption result for the fifth character is 193. The character -5 in the ciphertext is 193. Thus, the result of encryption for plain text with a length of 5 characters is 85, 238, 135, 219, and 193. Plaintext = 213 points, 110 points, 7, 91 points 65 points. The cipher text is 85, 238, 135, 219, and 193 with shift cipher 128 encryption (Ye and Ma, 2013).

Please take note of the decryption technique with shift cipher 128, which is described as follows. If the plain text is composed of elements in 256, the decryption process with shift cipher 128 is to add (modulo 256) each element by 128. Let us consider the example where the ciphertext has a length of 5 characters: 85, 238, 135, 219, 193. To decrypt the ciphertext with shift cipher 128, we need to perform the following steps. The first character in the ciphertext is 85. We add (modulo 256) 85-128, which gives us 85 +mod 256 128 = (213) mod 256 = 213.

Therefore, the decryption result for the first character in the ciphertext is 213 213 corresponding to the $(5\text{-}1)^{th}$ character in plaintext. Similarly, we can decrypt the other characters in the ciphertext. For example, the second character in the ciphertext is 238. We add (modulo 256) 238-128, which gives us 238 +mod 256 128 = (366) mod 256 = 110. Therefore, the decryption result for the second character in the ciphertext is 110 110 corresponds to the $(5\text{-}2)^{th}$ character in plaintext. The decryption process for the rest of the ciphertext characters can be done in the same manner. Thus, the decryption results for the ciphertext with a length of 5 characters: 85, 238, 135, 219, and 193 are 213, 110, 7, 91, and 65. In summary, the ciphertext is 85, 238, 135, 219, 193 and the plaintext obtained by decrypting it with shift cipher 128 is 213, 110, 7, 91, 65.

The discussion regarding the function of the custom Logistics map is as follows. In the process of creating the

Unimodular Matrix of Z_{256}, we often encounter the word "random" selection. This "random" selection process involves the Log map Custom function which is based on the logistics function. The Custom Log map function is a custom function with Input = 3 digit number (example: 230) and constant r and output = number with more than 180 digits. The Logistics function is a recursive function defined as:

$$f(n+1) = r \times f(n) \times (1 - f(n)), \, for \, n = 1, 2, 3, 4, 5 ... etc$$

Thus we can calculate $f(2)$, $f(3)$, $f(4)$, ..., f(1 million), but cannot calculate $f(3/2)$, $f(\pi)$, $f(-4)$ etc. Note that to calculate $f(n+1)$ we need the value of the constant r and to calculate $f(n)$ we need the value of $f(n-1)$ (Kordov, 2021). Some important things about our custom log map function algorithm are as follows. File python: pyeon_matriks_engine2.py. Lines 307-330. Function name: def logmap3 (vin initial value, vinR): Input: Vin initial value is input in the form of 3-digit numbers (example: 230) vinR is input constant r (example: 0, 3471). Output: Numbers totaling more than 180 digits (Muktyas *et al*., 2021). Please note that in the 308th line, the value of vin initial value entered by the user is modulated by 1000. The point is so that the vin initial value is in the range 0-999. Next, the 310th-316th line serves to reduce the vin initial value input by the user to 0,... (zero commas umpteenth). For example, if the user inputs the value vin Initial Value = 233, the value will be changed to 0.233. This value will be used as $f(0)$. Moreover, the 319th line sets calculation precision to 20 decimal places. Next, rows 321 and 322 calculate the logistic function based on $f(0)$ and the constant r = vinR to $f(20)$. Finally in lines 323-328, if the iteration is in the calculation of $f(10)$, $f(11)$, $f(12)$, to $f(20)$, then the calculation results are appended to one, and then the comma is removed so that it becomes a long series of numbers (Abderrahim *et al*., 2012).

Please note the following example. Let $f(10)$ = 0.9998 $f(11)$ = 0.1233333 f(12) = 0.6777754. If $f(10)$, $f(11)$ f(12) are appended it will return: 0,99980,12333330,6777754. Then if the comma is removed it will become 099980123333336777754. This algorithm will append the calculation results $f(10)$, $f(11)$, $f(12)$, to $f(20)$. That way the number of digits will be very large. Then, if the total number of digits from the append calculation $f(10)$, $f(11)$, $f(12)$-$f(20)$ is odd, then add a digit 1 behind so that the number of digits becomes even. 099980123333306777754 in the example above is 21 digits. Since the number of digits is odd, then the 1st digit is added at the end to become: 099980123333306777754 1. Thus, the number of digits is 22, which is even (Gupta *et al*., 2019).

In this study, we also created several supporting functions as follows. The pyeon_matriks_engine2.py file contains the Consolidated Encryption Technique support functions (Obaida *et al*., 2022) as follows. (1) arrinverse256 is an array that stores information on the inverse of the multiplication operation modulo 256. (2) The serialize2 function is used when storing a matrix in a text file. (3) The printm2 function is used to visualize matrices in the command line. (4) The inverse1V256a function is used to find the inverse of a matrix. The process is to perform the same series of elementary row operations on the reference matrix and identity matrix to convert the reference matrix into an identity matrix. The result of a series of elementary row operations on the identity matrix will make the identity matrix an inverse matrix. (5) The multiV256a function is used to multiply two matrices for the multiplication operation modulo 256. (6) The function obe2V256 is an elementary row operation of the second type (multiply by a constant) to the multiplication operation modulo 256. (7) The function obe3V256 is an elementary row operation of the third type (addition of a row by a multiple of another row) to addition and multiplication operations modulo 256. (8) The unimodular1V256 function is a function to create a unimodular matrix where the entries are elements in Z_{256}. (9) The logmap3 function is a custom log map function that is used to make random selections (Ojobor and Obihia, 2021). The following is a simple example for the implementation of the proposed algorithm, that will end this section.

```
=========================================
Now, insert your file: Video.mp4
=========================================
The 1-dimensional binary matrix of your file:
p = [0 0 0 ... 219 80 7] with size: 1×967617
Enter Password 1: 7
Enter Password 2: 77777
--------------------------
The encryption process begins.
--------------------------
Password 1 will be used as the size of Hill cipher's key matrix, that is (7×7) Password 2 will be used as the initial value of the logistic map. 77777 --> 0.777771
 The sequence of the logistic map generated by ×0 = 0.777771 is:
[46 80 179 196 107 156 17 12 160 95 205 206 95 84 112 156 182 183 76 181 186 82 177 202 120 143 60]

The upper triangular matrix key based on the logistics sequence formed:

[[1 46 80 179 196 107 156]
 [0 1 17 12 160 95 205]
 [0 0 1 206 95 84 112]
 [0 0 0 1 156 182 183]
 [0 0 0 0 1 76 181]
 [0 0 0 0 0 1 186]
 [0 0 0 0 0 0 1]]

We use elementary row operation to fill in the lower triangular matrix. Here is the key matrix:
```

[[1 46 80 179 196 107 156]
[82 189 177 98 104 165 197]
[177 206 81 145 227 79 76]
[202 76 32 63 68 36 207]
[120 144 128 232 225 116 213]
[143 178 176 253 124 198 222]
[60 200 192 244 240 20 145]]

The partition process begins.

Based on Password 1 and The Division Theorem, the plaintext matrix will be split into 967617 = 7×138231 + 0, such that (7×138231) and (1×0)
(7×138231) array part:
[[0 0 0 ... 47 196 153]
[138 197 98 ... 167 60 230]
[88 123 222 ... 13 152 60]
...
[3 119 198 ... 191 71 222]
[169 251 51 ... 154 82 30]
[185 123 63 ... 219 80 7]]

(1×0) array part:
[]

Hill cipher + Shift cipher process begins.

The (7×7)-size key matrix will be multiplied by the (7×138231)-size plaintext matrix and then proceed to the rest (1×0)-size with the Shift cipher 128:

Ciphertext (7×138231) part:
[[95 111 104 ... 175 109 4]
[44 2 76 ... 247 88 171]
[24 207 210 ... 5 60 228]
...
[132 250 53 ... 86 71 153]
[36 90 17 ... 121 85 112]
[253 127 159 ... 223 220 247]]

Ciphertext (1×0) part:
[]

Now, we will reshape the ciphertext matrix from (7×138231)-size into a 1-dimensional matrix again, (1×967617)-size + matrix from Shift cipher: (1×967617) array part:
[95 111 104 ... 223 220 247]
(1×0) array part:
[]
Finally, we will regroup two previous matrices and save them as binary files titled "Video_encrypted. mp4":
[95 111 104 ... 223 220 247]

The encryption process is finished.

Your encrypted file is in the same folder as the original file.
The encryption time is 0.49244189262390137 sec.

Results and Discussion

In this session, we will examine the research results obtained. After implementing a combination of Hill cipher, partition, shift cipher, and unimodular matrix logistic functions in the video encryption process, satisfactory results were obtained. By using a unimodular matrix on the Hill cipher, the complexity of the encryption key increases and provides a higher level of security. Then, by dividing the video block into several parts and doing a shift cipher on each part, it can increase resistance to attacks. Meanwhile, the use of the logistics function in each part of the video provides variations in each block thereby increasing the resistance of each block to attacks. In testing, this technique succeeded in producing well-encrypted videos and being able to maintain the original video quality properly. However, there is a weakness in this technique, namely the complexity of the algorithm is quite high, so it takes a long time to encrypt large videos. Therefore, in future research, it is possible to develop more efficient techniques to increase the encryption speed of large videos (Arifin *et al*., 2022a; Muktyas *et al*., 2021).

We will document the findings of this research throughout this session. The Combined Encryption Method is the method that will be applied in this study. Assume the following circumstances exist. Our file is more than 1 Kilo Bytes in size (KB). Keep in mind that 1,024 bytes make up 1 kilobyte. The file will be encrypted using the Encryption with Partition Method. 1,024 bytes are utilized as the partition size (1 KB). Keep in mind that the file size and the partition size must be less than each other. This encryption approach employs two different kinds of encryption: (1) Unimodular Matrix Encryption over Z_{256}. (2) Use Shift cipher 128 to encrypt. The results of the Unimodular Matrix encryption on Z_{256} are added with the results of the shift encryption (Rosalina, 2020). The front view of the digital video encryption and decryption application that we have developed can be seen in detail in Fig. 2.

The step-by-step use of the encryption-decryption application for the encryption process is as follows. Make sure you have installed Python and the required

DOI: 10.3844/jcssp.2023.847.860

packages on your computer/laptop. Make sure your computer/laptop has extracted the encryption and decryption application made using NodeJS. Make sure you have downloaded the source file that you want to encrypt, namely sample-10s.mp4. Make sure you have an encryption matrix. Open the encryption-decryption application. Click the open digital file button. Encrypt digital files. Select the sample-10s.mp4 file then click ok. Wait for the results to complete as shown in the command line window that opens. Please see the end of this chapter to know the encryption process algorithm (Taj *et al*., 2021). An illustration of this process can be seen in the following Fig. 3.

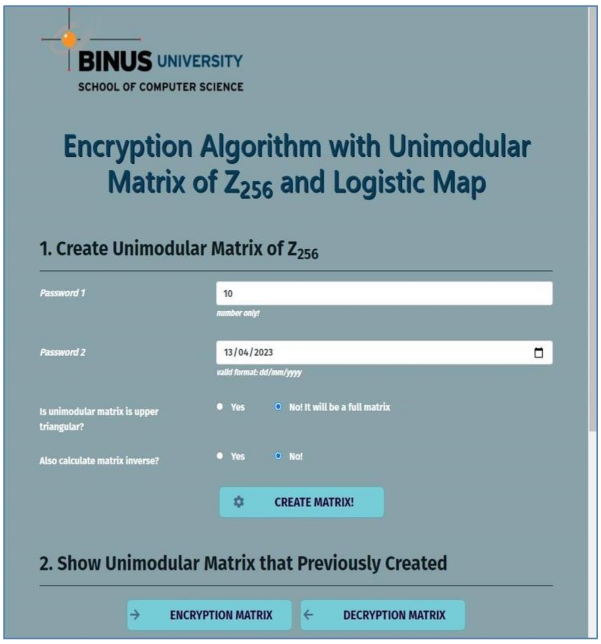

Fig. 2: Front view of the application made

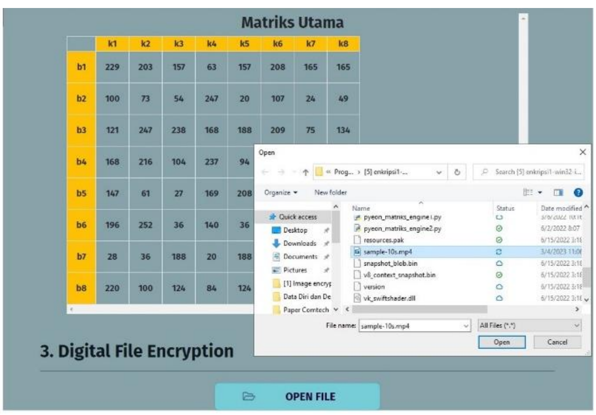

Fig. 3: The current view will open the video sample data for encryption

The encryption process algorithm that we apply is as follows. It is assumed here that we will encrypt and decrypt a video file named sample-10s.mp4. The encryption and decryption processes are handled by a Python file: Encryptp2.py. Lines 43-70 determine whether to encrypt or decrypt. It all depends on input from the user. If the chosen one is to perform the encryption process, then lines 76-92 will load the contents of the resulting matrix1qutama.dat file into the machine's memory as an encryption matrix. The encryption matrix has 1,032 entries. Thus, the encrypted source file, sample-10s.mp4 which is 5,485,983 bytes in size will be partitioned with each partition measuring 1,032 bytes. The partitioning process above results in 5,357 perfect partitions and 1 imperfect partition. This imperfect partition is 415 bytes in size as follows: (a) The 1^{st} perfect partition contains the 1st to $1,024^{th}$ bytes. (b) The 2^{nd} perfect partition contains the $1,025^{th}$ $2,048^{th}$ bytes. (c) ...and so on ..., moreover (d) The $5,356^{th}$ perfect partition contains 5,483,521 bytes up to the 5,485,544 bytes. (e) The $5,357^{th}$ perfect partition contains 5,484,545 bytes up to 5,485,568 bytes. (f) The imperfect partition contains 5,485,569 bytes to the 5,485,983 bytes.

Those 103^{rd} to 157^{th} rows will iterate over the 1st Perfect Partition to the $5,357^{th}$ Perfect Partition to multiply by the encryption matrix. Remember that the perfect partition is 1,024 bytes in size. All perfect partitions can be transformed into a matrix of 32 rows and 32 columns. The encryption matrix is a unimodular matrix over Z_{256} which has 32 rows and 32 columns. Perfect partition entries and unimodular matrices are elements in Z_{256}. So, we can multiply the unimodular matrix over Z_{256} and the perfect partition. (a) 1^{st} perfect partition encryption result = encryption matrix × 1^{st} Perfect Partition (as matrix) (b) 2^{nd} perfect partition encryption result = encryption matrix × 2^{nd} perfect partition (as matrix) (c) etc.

The results of this encryption are directly written to the output file (not stored in memory) so as not to burden memory performance. If the perfect partition encryption iteration has been completed, then the encryption process is continued by encrypting the imperfect partition using the shift cipher 128 methods. Lines 146-150 represent the process. The following is a verification of the encryption process we use. For example, we use the encryption matrix as below:

87;176;70;118;215;186;3;131;210;151;177;21;207;161;2
03;233;15
8;169;141;13;233;101;230;192;136;71;134;184;229;62;8
3;23
63;151;9;120;217;181;188;20;167;90;140;250;185;173;1
35;112;12
5;49;126;224;209;186;211;247;157;224;119;111;213;78;
191;79

DOI: 10.3844/jcssp.2023.847.860

63;48;87;238;111;240;129;98;188;130;12;31;142;170;16
8;208;143
;140;238;35;170;170;3;169;173;149;86;128;228;20;147;228
101;16;146;35;35;193;0;28;40;63;190;16;86;40;28;94;71
;93;123; 59;202;222;226;185;243;5;239;5;6;207;10;134
123;240;46;158;48;73;175;183;110;11;85;241;89;19;246
;127;41;8
;171;240;118;102;29;97;51;148;12;129;94;83;112;160
109;144;226;242;237;59;119;209;46;116;241;175;250;2
49;252;106
;221;5;185;42;4;240;103;27;91;106;100;28;219;121;144;29
132;64;168;232;132;88;79;205;83;84;249;41;219;241;10
1;93;191;
180;44;80;204;244;216;70;86;155;98;35;143;154;171;53
94;96;108;204;94;148;118;169;89;173;211;5;183;176;47
;175;169;
139;15;144;226;98;243;6;72;35;42;67;241;143;136;248
255;48;214;134;127;42;203;75;57;74;122;22;92;20;86;2
38;48;181
;57;52;208;205;239;27;24;12;243;47;50;63;124;54
81;208;74;154;209;182;197;69;222;24;159;35;253;71;16
5;31;120;
37;2;101;114;22;124;215;233;246;217;41;222;123;147;58
129;208;42;122;1;214;181;53;126;65;70;128;214;48;146
;37;82;7;
130;65;247;8;200;147;207;196;92;34;174;163;110;102
240;0;96;96;240;160;176;176;32;240;144;63;75;147;221
;210;148;
4;191;223;128;73;187;208;29;205;151;213;1;193;39;40
106;32;100;132;106;156;242;242;44;234;198;158;209;3
8;134;102;
92;70;180;228;237;56;39;3;2;161;149;101;77;213;157;211
2;160;84;244;2;172;106;106;252;130;110;230;146;181;2
48;39;177
;11;95;219;100;6;156;199;246;218;185;78;153;251;125;244
38;224;60;28;38;196;222;222;180;166;42;18;214;138;27
;116;183;
139;75;103;53;53;41;66;4;122;43;191;162;101;89;118
179;112;94;78;51;162;15;143;26;243;117;105;11;37;247
;242;179;
42;248;150;190;218;245;248;104;71;238;128;41;124;21;74
199;176;166;214;71;90;51;179;242;7;193;101;191;177;1
23;121;29
;243;96;96;75;76;247;213;215;88;177;33;211;199;16;148
44;192;56;248;44;200;28;28;168;44;116;196;140;52;188
;212;24;5
1;13;137;186;196;64;71;38;164;29;158;87;175;15;158
8;128;80;208;8;176;168;168;240;8;184;152;72;56;104;2
48;144;24
8;251;242;3;201;217;165;155;139;253;2;204;132;151;151
99;112;62;46;227;194;127;255;186;163;69;121;59;245;2
31;221;11
8;157;81;92;77;130;185;144;197;240;181;109;186;119;134
;91
109;144;226;242;237;30;145;17;166;45;107;247;21;187;
169;83;42
;147;159;31;160;39;166;144;64;109;232;222;190;196;4;48
62;96;44;140;62;212;214;214;132;190;82;218;174;50;23
0;66;92;1
94;74;74;66;255;62;23;65;51;123;145;69;133;20;103
240;0;96;96;240;160;176;176;32;240;144;208;112;144;4
8;16;224;
16;80;80;16;208;15;141;141;217;197;102;144;104;247;118
185;80;90;42;57;166;77;205;14;121;191;27;193;207;5;7;
130;71;3
5;163;7;203;186;143;176;174;88;27;98;149;111;19
197;16;82;226;69;174;201;73;246;133;83;127;45;35;33;
251;90;59
;231;103;251;111;50;64;173;160;195;177;148;21;188;178
23;176;198;246;151;58;195;67;82;87;241;85;143;225;13
9;41;30;2
33;205;77;41;165;102;192;136;68;200;108;25;45;162;71
252;192;88;24;252;168;44;44;8;252;36;52;220;228;76;4;
184;4;20 ;20;4;116;216;0;32;188;63;89;207;8;9;217
101;16;146;34;229;110;233;105;182;37;179;95;205;131;
65;91;154
;155;199;71;91;79;114;64;152;53;82;255;6;207;10;134
39;176;102;150;167;154;19;147;50;103;97;133;31;81;91
;25;62;21
7;125;253;25;213;6;192;8;23;166;56;6;85;155;167
209;208;74;154;81;182;69;197;222;145;231;227;153;11
9;61;239;5
0;47;171;43;239;19;170;64;56;97;10;136;147;227;89;97
235;240;142;254;107;114;167;39;170;43;125;145;3;173;
207;85;6;
21;169;41;85;161;174;192;232;27;206;88;33;38;150;147
49;208;10;90;177;246;37;165;30;241;135;3;249;23;29;1
43;242;20
7;203;75;143;51;106;64;56;193;202;136;179;82;85;144

If we open the sample-10s.mp4 file in the notepad++ application, we will get a chaotic display like in the following Fig. 4.

Make sure the notepad++ application has the HEX-editor plugin installed. If we click View in HEX, it will look like the Figs. 5-6.

It looks much more human though it's still confusing. The characters in columns 0, 1, 2, 3, to f, are hexadecimal. To translate hexadecimal characters to numbers from 0-255, you can use the table in reference (Jameel and Fadhel, 2022). Figure 7 for more details.

Please compare the Dec to the Hex column in the table above. Okay, let's continue to observe the appearance in the command line of the encryption process at the bottom of the following line. Notice the text in yellow in the following Fig. 8.

The yellow text indicates the 5,357th perfect partition. Remember that the 5,357th perfect partition contains the 5,484,545 bytes through the 5,485,568 bytes. Since one row of the matrix consists of 32 columns, then based on the second row of yellow text we can conclude:

Fig. 4: The chaotic display in the notepad++ if we open the sample-10s.mp4

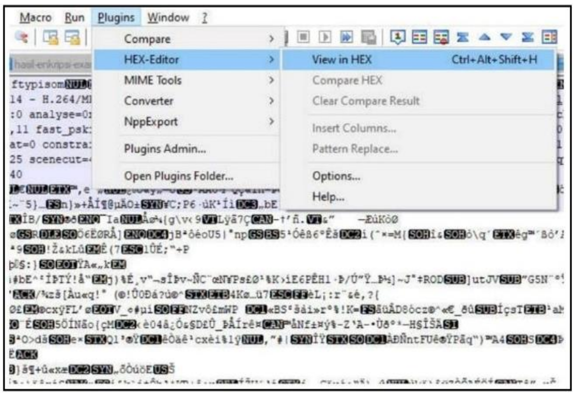

Fig. 5: The display in the notepad++ if we click to view in HEX

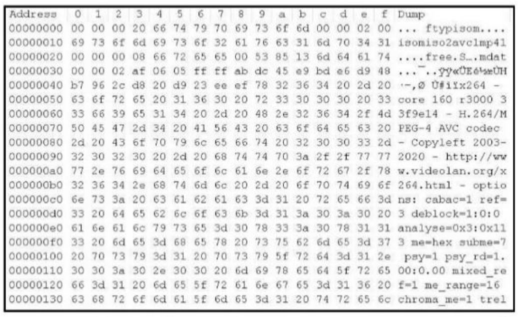

Fig. 6: Display of the characters in columns 0, 1, 2, 3, to f, are hexadecimal

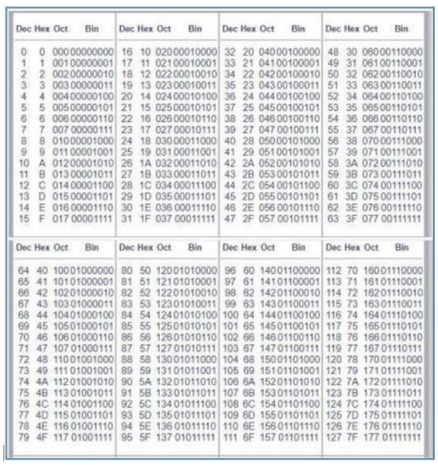

Fig. 7: The characters in columns 0, 1, 2, 3, to f, are hexadecimal translated to hexadecimal characters to numbers from 0-255

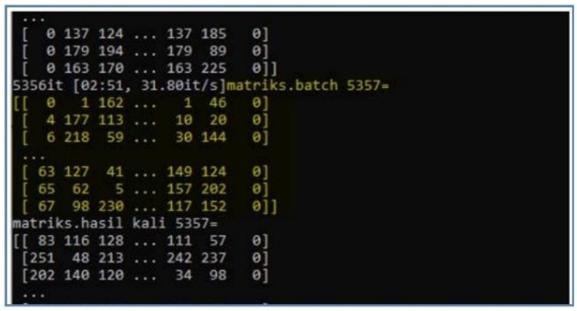

Fig. 8: The characters in columns 0, 1, 2, 3, to f, are hexadecimal translated to hexadecimal characters to numbers from 0-255

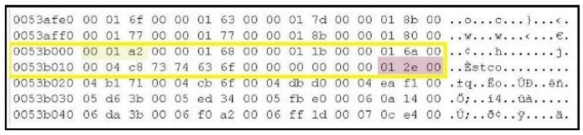

Fig. 9: The display of HEX-editor notepad++. If we scroll to addresses 0053b000 and 0053b010

(a) The 5,484,545 byte is 0. → HEX equivalent of 00 (b) The 5,484,546 byte is worth 1. → equivalent to HEX 01 (c) The 5th byte 5,484,547 is 162. → equivalent to HEX A2 (d) The 5,484,573 byte is worth 1. → equivalent to HEX 01 (e) The 5,484,573 byte is 46. → equivalent to HEX 2E and (f) The 5,484,573 byte is 0. → the HEX equivalent of 00. Please look at Fig. 7. Go back to HEX-editor notepad++. If we scroll to addresses 0053b000 and 0053b010, we will get a display like this in Fig. 9.

Notice the green and purple text in the yellow outline box. Is there any resemblance to HEX in the 6 bullet numbering above? Address 0053b000 in decimal is 5,484,544. That means address 0053b000 contains the 5,484,545 bytes to the 5,484,560 bytes. Thus we can conclude that the first 32 bytes of the 5,357th perfect partition are:

00, 01, A2, 00, 00, 01, 68, 00, 00, 01, 1B, 00, 00, 01, 6A, 00, 00, 04, C8, 73, 74, 63, 6F, 00, 00, 00, 00, 00, 00, 01, 2E, 00

which is equivalent to the following decimal:

0, 1, 162, 0, 0, 1, 104, 0, 0, 1, 27, 0, 0, 1, 106, 0,
0, 4, 310, 115, 116, 99, 111, 0, 0, 0, 0, 0, 0, 1, 46, 0

While the imperfect partition contains 5,485,569 bytes to the 5,485,983 bytes. That means, the incomplete partition is contained in the HEX address which is equivalent to 5,485,569 decimal places, which is 53b400. Figure 10 for more details.

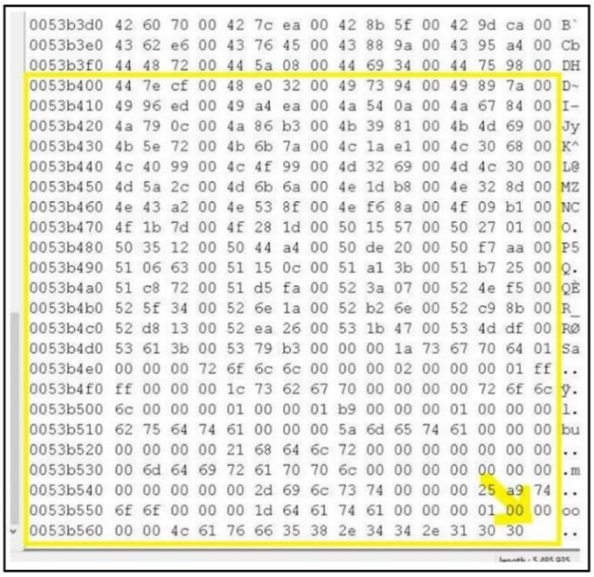

Fig. 10: The display of the incomplete partition is contained in the HEX address which is equivalent to 5,485,569 decimal places

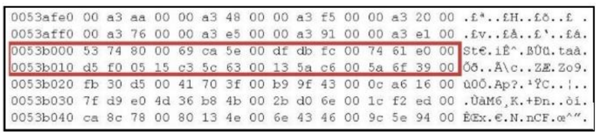

Fig. 11: The display of the incomplete partition is contained in the HEX address which is equivalent to 5,485,569 decimal places

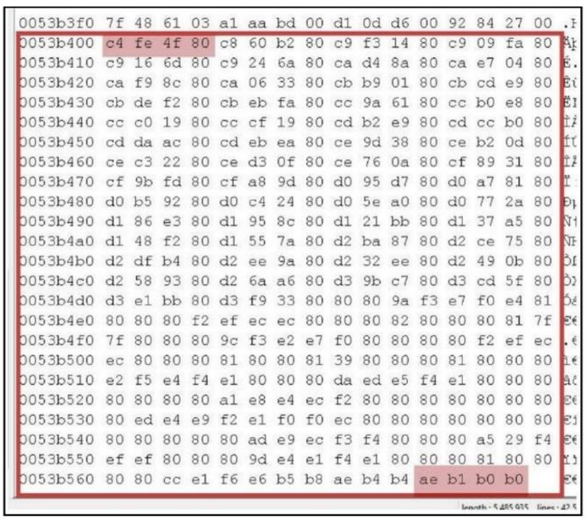

Fig. 12: The display of the beginning of an imperfect partition of 415 bytes

Note that this imperfect partition has a size of 415 bytes. Since 415 mod 16 = 1, it means that one cell in the last row is empty, as indicated by the yellow arrow in the bottom right corner. Switch to the encrypted file. The following are the 5,484,545 bytes to the 5,484,576 bytes that correspond to addresses 0053b000 and 0053b010. Figure 11 for more details.

In the red box above, please note the following HEX sequence:

53, 74, 80, 00, 69, CA, 5E, 00, DF, DB, FC, 00, 74, 61, E0, 00,
D5, F0, 05, 15, C3, 5C, 63, 00, 13, 5A, C6, 00, 5A, 6F, 39, 00

which is equivalent to the following decimal

83, 116, 128, 0, 105, 202, 94, 0, 223, 219, 252, 0, 116, 97, 224, 0, 213, 240, 5, 21, 195, 92, 99, 0, 19, 90, 198, 0, 90, 111, 57, 0

This is the first line of multiplying the encryption matrix with the 5,357th perfect partition. We turn to the HEX address 53b400 which is the decimal equivalent of 5,485,569. This is nothing but the beginning of an imperfect partition of 415 bytes. Figure 12 for more details.

Notice the 2 red squares above and below that contain HEX: C4, FE, 4F, 80 AE, B1, B0, B0. Note that C4 is equivalent to decimal 196, FE is equivalent to 254, 4F is equivalent to 79, 80 is equivalent to 128, AE is equivalent to 174, B1 is equivalent to 177 B0 is equivalent to 176. Also note that: (a) C4 is equivalent to decimal 196. If (196-128) mod 256 = 68 is equivalent to HEX 44, (b) FE is equivalent to decimal 254. If (254-128) mod 256 = 126 is equivalent to HEX 7E, c) 4F decimal equivalent 79. If (79-128) mod 256 = 207 HEX CF equivalent, (d) 80 decimal equivalent 128. If (128-128) mod 256 = 0 HEX 00 equivalent, (e) AE is 174 decimal equivalent. If (174-128) mod 256 = 46 HEX 2E equivalent, (f) B1 decimal equivalent 177. If (177-128) mod 256 = 49 HEX 31 equivalent (g) B0 decimal equivalent 176. If (176-128) mod 256 = 48 HEX 30 equivalent. Is there any resemblance to the contents of the yellow box below? This screenshot is a sample-10s.mp4 file that is opened using the HEX-editor notepad++ application starting at address 53b400. Figure 13 for more details.

The accompanying Table 1 displays the time needed for the encryption procedure the findings are somewhat unexpected. In general, the proposed method makes the encryption and decryption process take longer.

From the table, it can be concluded that the encryption time depends on the size of password 1. The larger the password 1, then the longer the time. This is because password 1 corresponds to the key size matrix in the hill cipher. Table 2 for more details.

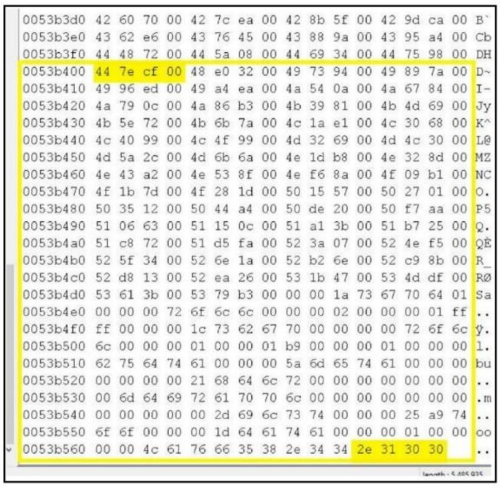

Fig. 13: The display of a sample-10s.mp4 file that is opened using the HEX-Editor notepad++ application starting at address 53b400

Table 1: Encryption time

Pass 1	Pass 2	Encryption time (seconds)
5	2345	422,7135.0000
26	2345	745.6350
100	2345	3378.7270

Table 2: Comparison of standard hill cipher and the proposed algorithm

Properties	Hill cipher standard	Proposed algorithm
Key matrix size of K_n	$n\leq4$ sually small n,	Any $n>0$
Key matrix storage of K_n	One whole matrix of K_n	Only 2 parameter

The decryption process is not similar to the encryption process. The difference is that the decryption process uses the inverse of the encryption matrix.

The main contribution to this topic is the development of a video encryption method that combines several cryptographic techniques such as Hill Cipher, partition, Shift Cipher, and unimodular matrix logistic functions. By using these techniques, video security will be increased and sensitive data on videos will be protected from unauthorized users. This method provides several quantitative advantages, including a higher level of security, and faster encryption times smaller file sizes compared to other video encryption methods. In addition, the use of a combination of different cryptographic techniques increases overall security, because the weaknesses of one technique can be compensated for by the other. In this case, the main contribution is the development of secure and effective video encryption methods by combining several existing cryptographic techniques. By using this method, it is hoped that video security can be improved and user privacy interests can be protected. In addition, this method also contributes to the development of the science of cryptography and its applications in multimedia, especially video.

Conclusion

Some conclusions that can be drawn from this study are that the use of a combination of several cryptographic methods can increase encryption security and reduce the possibility of attacks from irresponsible parties. The work motivation of this research is to increase security in the video encryption process which is increasingly important with the increasing use of video in various applications, such as in the world of business, media so on. By using a combination of several cryptographic methods, it is hoped that encryption security can be increased and prevent unauthorized access to encrypted videos.

The combination of several cryptographic methods used in this study can increase security in the video encryption process. Using partitions can speed up the video encryption process and reduce the computational burden on the device used. The application of unimodular matrices and logistic functions can increase the complexity of encryption, making it difficult for unauthorized parties to crack. The use of a combination of Hill cipher, partition, shift cipher, and unimodular matrix logistic functions in video encryption can provide a higher level of security than using a single cryptographic method. The results of this study can be used as a basis for developing more complex and secure video encryption techniques in the future.

Due to the challenge of locating a reversible matrix, the common hill cipher often utilizes a tiny K_n, $n\leq4$ key matrix. Furthermore, the complete K_n matrix is used as the key if $n>4$ with the conventional Hill Cipher. To address this issue, we build a Unimodular matrix in this study employing a unique logistic function as the key. $n>4$, yet it just requires two parameters (password 1-2). Encrypted files are more secure when Partition, Hill cipher shift cipher 128 are used together. The experimental findings indicate that the encrypted video is challenging for human eyes to decrypt. The program's slowness when encrypting files with big capacities is another flaw in this study. In this study, there is still an opportunity for future research, namely, to create a special function that maps audio-video into a matrix form so that it can be more real-time when performing the encryption process. Furthermore, in the future, it is still possible to combine several classic encryption methods that can be combined with the methods that have been successfully implemented, namely the hill cipher, shift cipher partition methods.

Acknowledgment

The authors would like to thank the reviewers for their informative comments, and suggestions ideas, which have helped mould this manuscript into something that is worthy of publication. This study is supported by the research and technology transfer office, Bina Nusantara University as a part of Bina Nusantara University's International Research Grant (PIB 2023) with contract number: 029/VRRTT/III/2023.

Funding Information

This study is supported and funded by Bina Nusantara University Research and Technology Transfer Office under the terms of the university's International Research Grant (PIB 2023) under contract number 029/VRRTT/III/2023.

Author's Contributions

Samsul Arifin, Wihikanwijna and Indra Bayu Muktyas: Coding the program, written, and finalized the manuscript.

Suwarno: Written and finalized the manuscript.

Muhammad Amien Ibrahim: Coding the program and simulating the data.

Felix Indra Kurniadi and Nerru Pranuta Murnaka: Simulating the data, tidying up the theoretical basis and the methods we use.

Ethics

This article is original and contains unpublished material. The corresponding author confirms that there is no conflict of interest in this study and no ethical issues involved.

References

Abderrahim, N. W. Benmansour, F. Z. & Seddiki, O. (2012). "Integration of chaotic sequences uniformly distributed in a new image encryption algorithm," *Int. Comput. Sci. Issues*, vol. *9*, no. 2, pp. 389-394.

Anton, H. (2018). *Elementary Linear Algebra*. John Wiley & Sons, Limited, 2018. https://books.google.co.id/books?id=ypROEAAAQBAJ

Arifin, S., & Garminia, H. (2019). Uniserial dimension of module zm × zn over Z using python. *Int. J. Sci. Technol. Res*, *8*, 194-9. https://www.researchgate.net/publication/334769797_Uniserial_Dimension_Of_Module_ZmxZn_Over_Z_Using_Python

Arifin, S., & Muktyas, I. B. (2018). Membangkitkan suatu matriks unimodular dengan python. *Jurnal Derivat: Jurnal Matematika Dan Pendidikan Matematika*, *5*(2), 1-9.

Arifin, S., Kurniadi, F. I., Yudistira, I. G. A., Nariswari, R., Murnaka, N. P., & Muktyas, I. B. (2022a, September). Image Encryption Algorithm through Hill Cipher, Shift 128 Cipher Logistic Map Using Python. In *2022 3rd International Conference on Artificial Intelligence and Data Sciences (AiDAS)* (pp. 221-226). IEEE.

Arifin, S., Muktyas, I. B., Al Maki, W. F., & Aziz, M. M. (2022b). Graph coloring program of exam scheduling modeling based on Bitwise coloring algorithm using Python. *Journal of Computer Science*, *18*(1), 26-32. https://doi.org/10.3844/jcssp.2022.26.32

Arifin, S., Muktyas, I. B., Prasetyo, P. W., & Abdillah, A. A. (2021). Unimodular matrix and bernoulli map on text encryption algorithm using python. *Al-Jabar: Jurnal Pendidikan Matematika*, *12*(2), 447-455. https://doi.org/10.24042/ajpm.v12i2.10469

Basavaiah, J., Anthony, A. A., & Patil, C. M. (2021). Visual Cryptography Using Hill Cipher and Advanced Hill Cipher Techniques. In *Advances in VLSI, Signal Processing, Power Electronics, IoT, Communication and Embedded Systems: Select Proceedings of VSPICE 2020* (pp. 429-443). Springer Singapore. https://doi.org/10.1007/978-981-16-0443-0_34

Delmi, A., Suryadi, S., & Satria, Y. (2020). Digital image steganography by using edge adaptive based chaos cryptography. In *Journal of Physics: Conference Series* (Vol. *1442*, No. 1, p. 012041). IOP Publishing. https://doi.org/10.1088/1742-6596/1442/1/012041

Dooley, J. F. (2018). The Machines Take Over: Computer Cryptography. In: History of Cryptography and Cryptanalysis. History of Computing. *Springer, Cham*. https://doi.org/10.1007/978-3-319-90443-6_10

Elkamchouchi, H., Salama, W. M., & Abouelseoud, Y. (2020). New video encryption schemes based on chaotic maps. *IET Image Processing*, *14*(2), 397-406. https://doi.org/10.1049/iet-ipr.2018.5250

Gupta, R., Pachauri, R., & Singh, A. K. (2019). Image encryption method using dependable multiple chaotic logistic functions. *International Journal of Information Security and Privacy (IJISP)*, *13*(4), 53-67. https://doi.org/10.4018/IJISP.2019100104

Hanson, R. (1982). Integer matrices whose inverses contain only integers. *The Two-Year College Mathematics Journal*, *13*(1), 18-21. https://www.tandfonline.com/doi/abs/10.1080/00494925.1982.11972572?journalCode=ucmj19

Hussein, M. K., & Amintoosi, H. (2023). Protection of images by combination of vernam stream cipher, AES and LSB steganography in a video clip. *Bulletin of Electrical Engineering and Informatics*, *12*(3), 1578-1585. https://doi.org/10.11591/eei.v12i3.4039

Ibrahim, A. K., Hagras, E. A., Mohamed, A. N. F., & El-Kamchochi, H. A. (2021, July). Chaotic isomorphic elliptic curve cryptography for secure satellite image encryption. In *2021 International Telecommunications Conference (ITC-Egypt)* (pp. 1-7). IEEE. https://doi.org/10.1109/ITC-Egypt52936.2021.9513949

Jameel, E. A., & Fadhel, S. A. (2022). Digital Image Encryption Techniques: *Article Review*. https://doi.org/10.47577/technium.v4i2.6026

Jarjar, M., Najah, S., Zenkouar, K., & Hraoui, S. (2020, April). Further improvement of the HILL method applied in image encryption. In *2020 1st International Conference on Innovative Research in Applied Science, Engineering and Technology (IRASET)* (pp. 1-6). IEEE. https://doi.org/10.1109/IRASET48871.2020.9092046

Komosko, L., Batsyn, M., Segundo, P. S., & Pardalos, P. M. (2016). A fast greedy sequential heuristic for the vertex colouring problem based on bitwise operations. *Journal of Combinatorial Optimization*, *31*, 1665-1677. https://doi.org/10.1007/s10878-015-9862-1

Kordov, K. (2021). Text encryption algorithm for secure communication. *International Journal of Applied Mathematics*, *34*(4), 705.

Muktyas, I. B., & Arifin, S. (2018). Semua Subgrup Siklik dari Grup (Zn, +). *Teorema: Teori dan Riset Matematika*, *3*(2), 177-186. https://doi.org/10.1109/AiDAS56890.2022.9918696

Muktyas, I. B., Sulistiawati, & Arifin, S. (2021, April). Digital image encryption algorithm through unimodular matrix and logistic map using Python. In *AIP Conference Proceedings* (Vol. *2331*, No. 1, p. 020006). AIP Publishing LLC. https://doi.org/10.1063/5.0041653

Obaida, T. H., Jamil, A. S., & Hassan, N. F. (2022). A Review: Video Encryption Techniques, Advantages and Disadvantages. *Webology 19*(1). https://www.webology.org/data-cms/articles/20220308035845pmwebology%2019%20(1)%20-%2098%20pdf.pdf

Ojobor, S. A., & Obihia, A. (2021). Modified Variational Iteration Method for Solving Nonlinear Partial Differential Equation using Adomian Polynomials. *Mathematics and Statistics*, *9*(4), 456-464. https://doi.org/10.13189/ms.2021.090406

Oliphant, T. E. (2007). Python for scientific computing. *Computing in Science & Engineering*, *9*(3), 10-20. https://doi.org/10.1109/MCSE.2007.58

Paragas, J. R., Sison, A. M., & Medina, R. P. (2019, June). Hill cipher modification: A simplified approach. In *2019 IEEE 11th International Conference on Communication Software and Networks (ICCSN)* (pp. 821-825). IEEE. https://doi.org/10.1109/ICCSN.2019.8905360

Rahman, M. N. A., Abidin, A. F. A., Yusof, M. K., & Usop, N. S. M. (2013). Cryptography: A new approach of classical hill cipher. *International Journal of Security and its Applications*, *7*(2), 179-190. https://www.earticle.net/Article/A210948

Rajvir, C., Satapathy, S., Rajkumar, S., & Ramanathan, L. (2020). Image encryption using modified elliptic curve cryptography and Hill cipher. In *Smart Intelligent Computing and Applications: Proceedings of the Third International Conference on Smart Computing and Informatics, Volume 1* (pp. 675-683). Springer Singapore. https://doi.org/10.1007/978-981-13-9282-5_64

Rihartanto, R., Ningsih, R. K., Gaffar, A. F. O., & Utomo, D. S. B. (2020). Implementasi vigenere cipher 128 dan rotasi bujursangkar untuk pengamanan teks. *Jurnal Teknologi dan Sistem Komputer*, *8*(3), 201-209.

Rosalina, N. H. (2020). An approach of securing data using combined cryptography and steganography. *International Journal of Mathematical Sciences and Computing (IJMSC)*, *6*(1), 1-9.36) https://doi.org/10.5815/ijmsc.2020.01.01

Suresh, P., & Ratheesh, T. K. (2020). A Data Security Scheme for the Secure Transmission of Images. In *Intelligent Data Communication Technologies and Internet of Things: ICICI 2019* (pp. 147-154). Springer International Publishing. https://doi.org/10.1007/978-3-030-34080-3_17

Taj, A. M., Abouhilal, A., Taifi, N., & Malaoui, A. (2021). Embedded Electronics Applied in Remote Laboratories Using NodeJs. *Iraqi Journal of Science*, 1-6. https://doi.org/10.24996/ijs.2021.SI.1.1.

Yang, P., Xiong, N., & Ren, J. (2020). Data security and privacy protection for cloud storage: A survey. *IEEE Access*, *8*, 131723-131740. https://doi.org/10.1109/ACCESS.2020.3009876

Ye, R., & Ma, Y. (2013). A Secure and Robust Image Encryption Scheme Based on Mixture of Multiple Generalized Bernoulli Shift Maps and Arnold Maps. *International Journal of Computer Network & Information Security*, *5*(7). https://doi.org/10.5815/ijcnis.2013.07.03