REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITE FERHAT ABBAS SETIF-1
FACULTE DE TECHNOLOGIE

THESE

Présentée au Département d'Electronique
Pour l'obtention du Diplôme de

DOCTORAT LMD

Filière : Electronique

Option : Instrumentation Electronique

Par

MOKHNACHE ABDELAZIZ

THEME

Implémentation d'Algorithmes Avancés de Traitement de l'Information Dédiés au Cryptage et à la Cryptanalyse

Soutenue le 16 / 07 / 2023 devant le Jury:

Pr. BOUKEZZOULA Naceur-Eddine	Univ. Sétif 1	Président
Pr. ZIET Lahcene	Univ. Sétif 1	Directeur de thèse
Pr. FERHAT HAMIDA Abdelhak	Univ. Sétif 1	Membre
Pr. BENCHEIKH Abdelhalim	Univ. Sétif 1	Membre
Dr. BEKKOUCHE Tewfik	Univ. BBA	Membre
Dr. SOUKKOU Ammar	Univ. Jijel	Membre

Publications and Conferences

Publications

Mokhnache, A., Ziet, L. (2020). *Cryptanalysis of a pixel permutation based image encryption technique using chaotic map*. Traitement du Signal, Vol. 37, No. 1, pp. 95-100. https://doi.org/10.18280/ts.370112

Conferences

Security Enhancement of an Image Encryption Scheme Using the Sine-Sine Chaotic Map and an Efficient Permutation Technique. Conference, Abdelaziz Mokhnache, Ziet Lahcene, Hafssa Mounir, SÉMINAIRE INTERNATIONAL SUR L'INDUSTRIE ET LA TECHNOLOGIE, Oran, Algeria on 14/03/2021

Index

Table of Figures

List of Tables

List of Abbreviations

AES: Advanced Encryption Standard

CBC: Cipher Block Chaining

CFB: Cipher Feedback

DCT: Discrete Cosine Transform

DES: Data Encryption Standard

DFT: Discrete Fourier Transform

ECB: Electronic Code Book

FRFT: Fractional Fourier Transform

GCM: Galois Counter Mode

GIMP: Gnu Image Manipulation Program

IBM: International Business Machines

JPEG: Joint Photographic Experts Group

MAC: Message Authentication Code

NIST: National Institute of Standards and Technology

NPCR: Number of Pixels Change Rate

OFB: Output Feedback

OTP: One Time Pad

PNG: Portable Network Graphic

RSA: Ronald Rivest, Adi Shamir and Leonard Adleman algorithm

UACI: Unified Average Changing Intensity

Acknowledgments

My thanks are wholly devoted to **ALLAH** for guiding me and helping me all the way to conclude this thesis successfully.

I would like to thank my supervisor – **Pr. ZIET Lahcene** for his valuable supervision, support and guidance during the course of my PhD degree.

Additionally, I would like to express my gratitude to **Dr. BEKKOUCHE Tewfik** for his guidance at the beginning of my journey, which was very helpful.

I also thank **Pr. BOUKEZZOULA Naceur-Eddine, Pr. FERHAT HAMIDA Abdelhak, Pr. BENCHEIKH Abdelhalim, Dr. BEKKOUCHE Tewfik and Dr. SOUKKOU Ammar** who honored me with their acceptance to examine this work.

I would like to thank my friends, lab mates, colleagues and research team for the precious time spent together.

My appreciation also goes out to my beloved family for their encouragement, patience and support through all the years of my studies.

ملخّص

تقدم هذه الرسالة مساهمتين في مجال أمن المعلومات، المساهمة الأولى هي اقتراح طريقة جديدة في تحليل الشفرات لكسر حماية إحدى الخوارزميات المنشورة حديثا والتي تدعي امتلاك مفتاح ذو حجم لانهائي. تستخدم خوارزمية تحليل الشفرات المقترحة مزيجًا من هجمات النص العادي المختار وهجمات البحث المعمق لكسر الخوارزمية المنشورة تمامًا.أما المساهمة الثانية فهي مقارنة تحليلية بين إحدى خوارزميات تشفير الصور الحديثة وخوارزمية AES المعروفة. استنادا على النتائج المتوصل إليها من خلال المساهمتين السابقتين، أُبرزت في الأخير بعض المشكلات الخطيرة والشائعة في العديد من أنظمة تشفير الصور الحديثة في شكل قائمة من الأنماط التي ينبغي تجنبها، هدفها لفت انتباه مصممي هذه الخوارزميات إلى تلك الأخطاء الشائعة، أملا في تحسين جودة أنظمة التشفير المستقبلية.

الكلمات المفتاحية: أمن البيانات، تشفير الصور ، التشفير ، تحليل الشفرات ، نظرية الفوضى

Abstract

In this thesis, we present two contributions in information security field. The first contribution is the proposition of a new hybrid cryptanalysis method against a new published algorithm claiming infinite key size. The proposed cryptanalysis algorithm uses a combination of chosen-plaintext and brute-force attacks to completely break the proposed algorithm. The second contribution is a comparative analysis between a pixel-wise encryption schemes and AES algorithm in the context of a real web application. Based on the results of the work above, a highlight on some common major issues found in many modern image encryption schemes is done at the end. It presents a curated list of image encryption anti-patterns aiming to draw algorithm designers' attention to those repeatedly committed errors and help improving the overall quality of future cryptosystems.

Keywords: Security, Image Encryption, Cryptography, Cryptanalysis, Chaos

Résumé

Dans cette thèse, nous présentons deux contributions dans le domaine de la sécurité de l'information. La première contribution est la proposition d'une nouvelle méthode de cryptanalyse hybride contre un nouvel algorithme publié revendiquant une taille de clé infinie. L'algorithme de cryptanalyse proposé utilise une combinaison d'attaques de texte en clair choisi et de force brute pour casser complètement l'algorithme proposé. La deuxième contribution est une analyse comparative entre un schéma de chiffrement pixel par pixel et un algorithme AES dans le contexte d'une application Web réelle. Sur la base des résultats des travaux ci-dessus, une mise en évidence de certains problèmes majeurs communs rencontrés dans de nombreux schémas de cryptage d'image modernes est effectuée à la fin. Il présente une liste organisée d'anti-modèles de chiffrement d'image visant à attirer l'attention des concepteurs d'algorithmes sur ces erreurs commises à plusieurs reprises et à aider à améliorer la qualité globale des futurs cryptosystèmes.

Mots-clés : Sécurité, Cryptage d'images, Cryptographie, Cryptanalyse, Chaos

Introduction

Since the early days of humanity, information security was known to be important [1]. Human beings are known to have a great sense of privacy. Therefore, they tend not to share everything in public. They usually share information with someone and hide it from someone else. This is why cryptography was invented.

Cryptography is the science that study secure means of communication in the presence of unwanted third parties. Its origins dated to thousands of years ago and with time it became increasingly important [2]. Nowadays, billions of people are connected to the internet, their devices are sharing and receiving billions of gigabytes of data every day over the wires of internet. With this increase of connectivity, the demand for secure ciphers that can ensure the security of all that data increased as well. Therefore, fast and secure ciphers were developed over the last few decades and the world took security and cryptography very seriously [3].

Unlike two decades ago where information on the internet were just text and very few images, nowadays multimedia represents the largest portion of transferred data. With the increasing popularity of social networks and video streaming services, researchers and companies started looking for new means of data compression and encryption techniques that could handle the increasing load of transmitted information. Some researchers started pivoting from studying general purpose ciphers and began developing new specialized techniques of cryptography dedicated to multimedia such as images and videos. In the recent years, the number of researches treating the topic of image encryption has increased significantly.

The newly proposed image ciphers differed in many aspects and metrics. However, they focused mainly on security and performance enhancement. Some of them improved encryption and decryption speed. The other focused on enhancing the overall security while another part tried to achieve both of these objectives and develop faster and more secure schemes. However, every system has weaknesses. Some new ciphers were unsecure and got cracked by cryptanalysts just few months or a couple of years after being proposed. Cryptanalysis is crucial for propelling cryptography. It draws the attention of cryptosystem designers to the vulnerabilities found in their systems and prevents weak ciphers from being adopted and used in sensitive applications. In summary, cryptanalysis makes ciphers more secure [4].

In this context, we proposed a hybrid method of cryptanalysis that was applied on a recently designed image cipher claiming to be secure and has infinite key space. With a combination of

differential and brute-force attacks, we developed an algorithm capable of fully identifying and completely revealing any image that was encrypted with that cipher.

Motivated by the popularity of chaos based encryption systems, we tried to implement a chaos based encryption algorithm into a real world image sharing web application. A comparative analysis was performed between a recently proposed chaos based image encryption scheme claiming to be fast and the Advanced Encryption Standard (AES) based on the real needs and requirements of the application.

During the time that we've spent analyzing new proposed image ciphers, we found some anti-patterns that are commonly committed by many image cryptosystem designers. The main reason for these common issues based on our analysis is that many cipher designers don't usually think about the engineering context where the cipher is going to be implemented in real world applications. Some of the vulnerabilities are security related while others are performance and usability issues. We collected those common issues and highlighted them to help cryptosystem designers proposing more secure and performant algorithms by avoiding these anti-patterns in future systems.

This thesis is going to be organized into five chapters. The first one will be a state of the art about cryptography in general. Different types of cryptosystems will be presented and explained. Moreover, to help the reader get prepared for the cryptanalysis part later, a section will be dedicated to talk about attacks of different types that are relevant to modern ciphers.

Since our main work is related to digital image security, the second chapter will be a state of the art about image encryption. It will start by an introduction to the notion of digital images and basic digital image processing techniques. Then an overview on image ciphers is taken, focusing on the classification of modern image encryption techniques and the most used evaluation metrics.

The third chapter will be dedicated to the cryptanalysis method that we have proposed on the a newly published image encryption algorithm. First, the published encryption scheme is briefly described. Then, the attack algorithm is explained in details and the practical results are presented and discussed. At the end of the chapter, the vulnerabilities that was found in the scheme that led to its crack are highlighted.

In the fourth chapter, a comparative analysis between a pixel-wise chaos based image encryption algorithm and AES is conducted in the context of a web application. A customized

test suite is designed according to the requirements of the application, then both the chaos based algorithm and AES are compared together to select the most suitable encryption algorithm for that type of applications.

Based on our experience trying to implement a pixel-wise encryption algorithm into a web application, the fifth chapter will be a summary of our findings about some common vulnerabilities and issues found across multiple modern image encryption schemes that follow the pixel-wise encryption model. These anti-patterns are presented and explained in order to help cryptosystem designers avoid them when designing future ciphers.

Chapter 1

Introduction to Cryptography

Index

Chapter 1

Introduction to Cryptography

1.1 Introduction

Cryptography is the science of securing data and communications stored or sent over a public channel. It is generally the study of techniques that turns confidential data of arbitrary length into codes that looks random using a small piece of information called a secret key. The secured data can be recovered only if the secret key is known.

Humanity knew the importance of cryptography since the early days. However, the need for it increased with the technological advancement and the numerical transition that the world knew in the last decades.

This chapter will be an overview on cryptography. The start will be some historical information about its origins and evolution. Then its objectives and categories are presented. Finally, we will have a look on cryptanalysis which is the art of breaking cryptography. Cryptanalysis represents the core of our first contribution in this thesis.

1.2 History

One of the earliest ciphers that was used on a large scale was a transposition method of encryption known as the skytale cipher, used by the Spartans around 500 B.C. The skytale was a strip of leather wound around a wooden rod like a belt. The message was written on the strip and the strip was unwound to scramble the message. On the other side, the original message is recovered by winding the strip again around a rod of the same diameter of the one used in encryption [1]. That was the same principle of nowadays modern symmetric cryptography.

Then, the great emperor of Rome Julius Caesar, used a cipher where every alphabet was shifted three places to the left. Hence, D will be written in place of A and B will be replaced with E. This cipher was used for confidential communication between Roman Generals and the emperor. The cipher was named Caesar cipher after the name of the emperor [2].

In 1586, Blaise de Vigenère came up with an innovative new cipher. He developed a powerful new way of encryption by the use of twenty-six distinct cipher alphabets to encrypt a message. The cipher alphabets are all simple Caesar shifts of one to twenty-six arranged in a square. Vigenere cipher was thought undecipherable until 1863 when Friedrich Kasiski published a general method to decipher it [1].

In 1917, the american Edward Hebern invented the electro-mechanical machine in which the key is embedded in a rotating disc. It encodes a substitution table that is changed every time a new character is typed [1].

In 1918, the German engineer Arthur Scherbius invented the Enigma machine for commercial use. Rather than the one rotor used by Hebern's device, it uses 3 rotors. The Enigma machine was relied upon heavily by the German military in World War II [1].

After the start of World War II in 1939, Allan Turing and a team of cryptographers at Bletchley Park in England, invented an electro-mechanical device called the Bombe. It was used to decipher the Enigma-encrypted communication between the German Navy headquarters at Kiel and the submarines in the North Atlantic [1].

In 1945 Claude E. Shannon published an article called "A mathematical theory of cryptography" where he set the fundamentals for modern cryptography [3].

In 1970s, IBM designed a block cipher to protect its customers' data. It was a fairly complicated set of bit operations performed in 16 rounds on the 64 bits of a message and 56 bits of key size. It was ready to be run by standard digital computer hardware at great speed. In 1977, the US adopted it as a national standard called the Data Encryption Standard, or DES [4]. This cipher was widely used in computers and in a lot of other devices and applications. It remained in use until it was broken in 1997.

In 1976, a protocol for sharing secret keys through a public channel was proposed by Diffie and Hellman [5]. Their revolutionary idea opened the door for a new type of cryptosystems known later by public-key cryptography.

In 1978, Rivest, Shamir and Adleman proposed the first public-key algorithm called RSA exploiting a mathematical problem named “factoring problem” [6]. RSA remained one of the most important public-key algorithms until now.

In June 1997, a group of scientist broke DES for the first time. The group won the challenge proposed by RSA Security company to highlight the lack of security provided in Data Encryption Standard.

In September 1997, NIST (National Institute of Standards and Technology) announced a call for new algorithms in order to select the Advanced Encryption Standard, or AES the successor of DES in a more open and transparent way. AES got selected and standardized in 2001.

1.3 Objectives of Cryptography

A trustworthy cryptosystem has to abide by certain rules and objectives. Any cryptosystem that fulfils the objectives mentioned below is considered safe and hence can be utilized for cryptographic properties.

1.3.1 Confidentiality

The first objective of cryptography and the most intuitive one is confidentiality. Which means that no one besides the intended recipient can understand the message or the information that was sent. Confidentiality was the primary goal that pushed the invention and development of cryptography in the first place.

1.3.2 Integrity

A cryptosystem has to ensure that the information in storage or in transit between the sender and the recipient cannot be altered by any means without being detected. The receiver of a message should be able to check whether the message was modified during transmission, either accidentally or deliberately.

1.3.3 Authentication

Authentication means that the receiver of a message should be able to verify its origin. When initiating a communication, the sender and the recipient should be able to identify each other identity, and no one should be able to send a message to the recipient and pretend to be the already authenticated sender.

1.3.4 Non-repudiation

This property assures that the sender can never convincingly deny his intention to send the message. The importance of this property appears when the sender denies his sending of the message and the receiver want to prove him wrong in front of a third party judge.

1.4 Types of cryptosystems

Cryptosystems can be divided into multiple categories depending on which criteria is considered. However, they are generally classified into two categories based on the encryption key nature: symmetric and asymmetric ciphers.

1.4.1 Symmetric ciphers

Symmetric ciphers are the most convenient and intuitive type of ciphers to think about. It is a system of cryptography composed of two algorithms: an encryption algorithm and a decryption one, as well as a shared secret key between two parties or more that want to share data securely over a public channel. The data encrypted with the encryption algorithm using a given secret key can be fully decrypted with the decryption algorithm using the same key exclusively [2]. This process is illustrated in Figure 1-1.

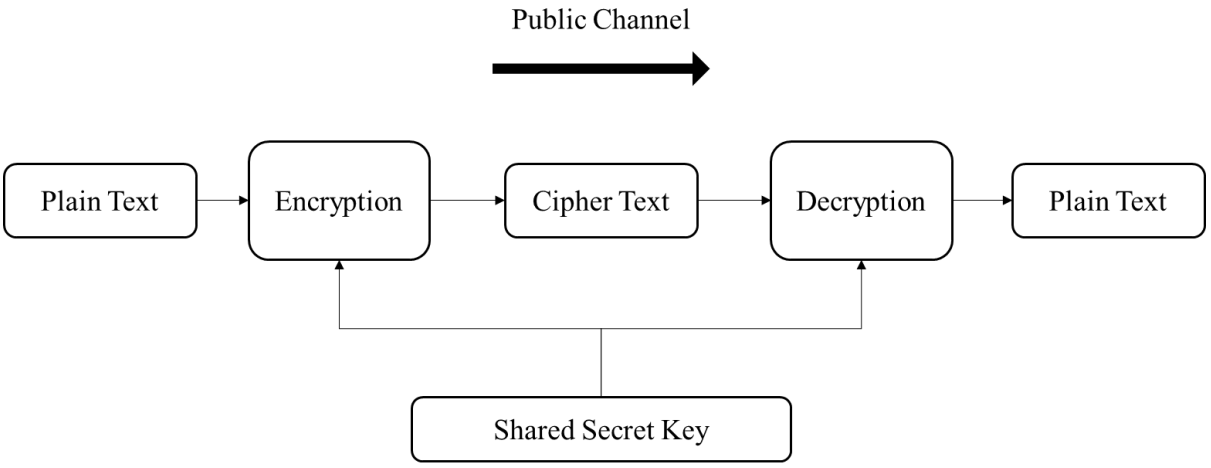


Figure 1-1: Symmetric ciphers principle

Both encryption and decryption algorithms implementations need to be completely known to the public and should not involve any hidden parts. The secrecy of the plain data given the cipher data should be only dependent of the secrecy of the secret key [7]. This is a well-known principle in cryptography called Kirchhoff's' principle.

Sometimes in symmetric ciphers, the encryption and decryption keys can be slightly different from each other. However, one of them should be easy to compute given the other one [7].

Based on the data type fed into a symmetric cipher, we can distinguish 2 types of ciphers: stream ciphers and block ciphers.

1.4.1.1 Stream ciphers

Stream ciphers are encryption algorithms that encrypt streams of data bit by bit or character by character using a pseudo random generator and a secret key.

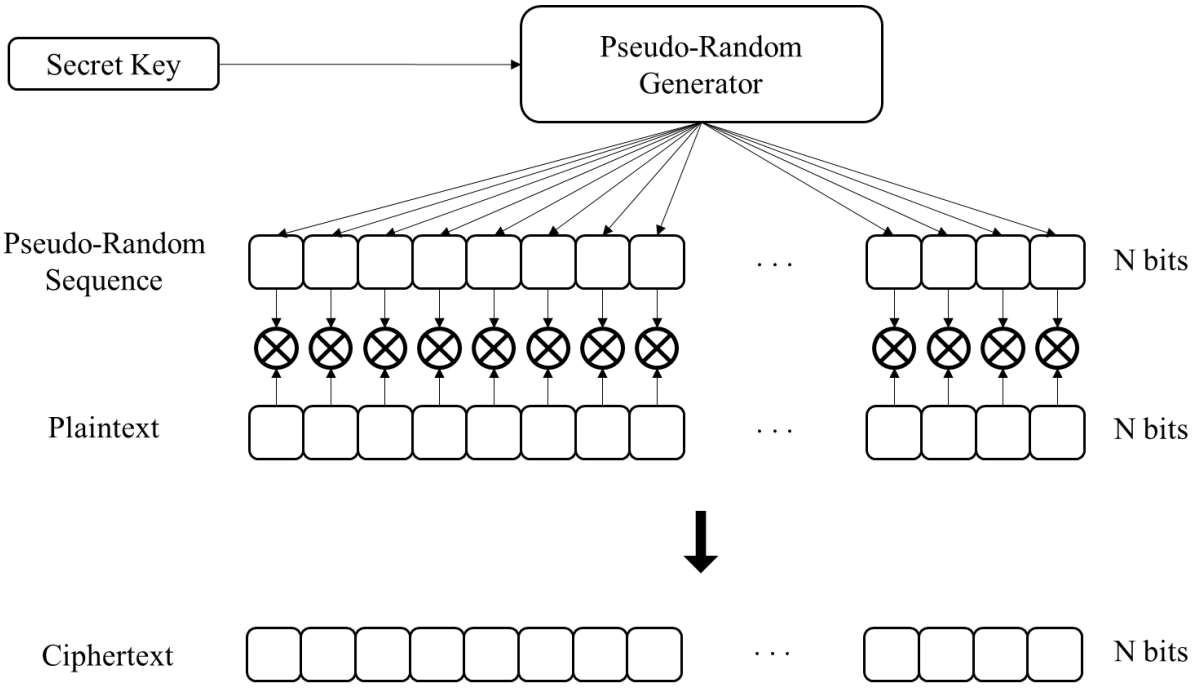


Figure 1-2: Stream cipher

Figure 1-2 illustrates how stream ciphers work in general. Generally, they work by adding a bit from a pseudo-random sequence to the corresponding plaintext bit modulo 2 (XOR operator). The key stream is generated using a cryptographically secure pseudo-random bit generator. Since the operation of addition modulo 2 is reversible, the decryption on the other hand is done by generating the same pseudo-random sequence and adding it bit by bit to the cipher text.

There are synchronous stream ciphers where the key stream depends only on the key, and asynchronous ones where the key stream also depends on the resulting ciphertext.

Stream ciphers are known to be very fast and resource efficient [4]. They are mostly used in applications and environment with little computational resources such as cell phones and embedded devices [7].

Some examples of the most used stream ciphers in real applications are: ChaCha, Salsa20, RC4 …etc.

1.4.1.1.1 The One Time Pad

The one-time pad (OTP) is an encryption technique where a plaintext is paired with a truly random secret key having the same length. Then, each bit or character of the plaintext is encrypted by combining it with the corresponding bit or character from the pad using modular addition as illustrated in Figure 1-3. The one-time pad algorithm was invented by Gilbert Vernam in 1917.

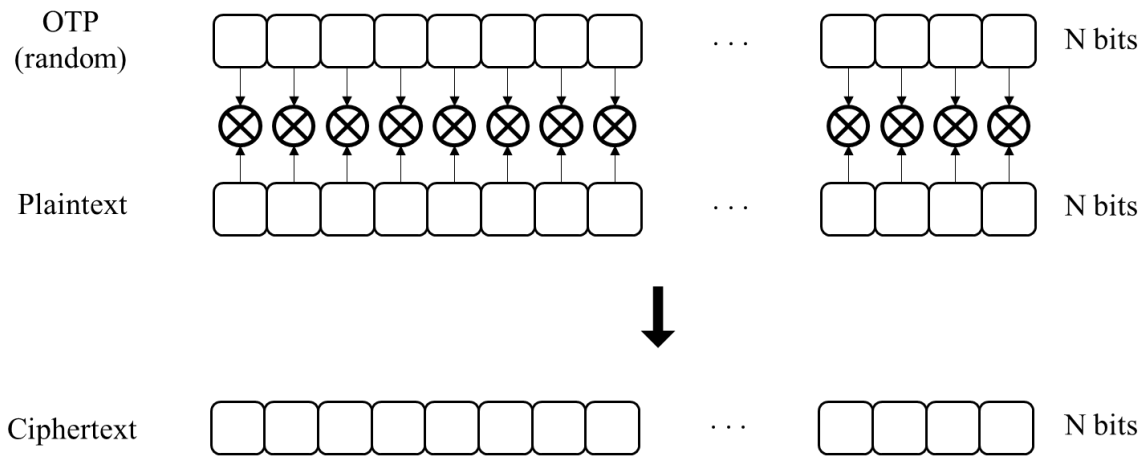


Figure 1-3: Encryption using One-time Pad algorithm

The one-time pad was proven by Shannon to be perfectly secret in 1949 [8]. Perfect secrecy means that the ciphertext leaks no information about the plaintext. Therefore, the algorithm resists all ciphertext-only attacks. Even with unlimited computing power an adversary will not be able to get any information about the plaintext.

However, one-time pad and all perfectly secret ciphers are usually impractical. It is hard in practice to generate truly random bit sequences of sufficient length and transmit them securely to the recipient on every encryption operation.

Although the impracticality of the one-time pad in real world, it was the building block for a lot of other algorithms that came after. Those algorithms abandoned the unrealistic assumption of unlimited computing power and considered the feasible attacks in practice only. Truly random bits' sequences were replaced by pseudo-random ones generated by a good pseudo-random bit's generator.

1.4.1.2 Block ciphers

Block ciphers on the other hand doesn't encrypt one bit of plaintext at a time but take a block of data instead. As illustrated in Figure 1-4, block ciphers are algorithms that take as inputs a block of fixed length of plaintext and a key and output a block of ciphertext having the same size as the plaintext. A good block cipher should have good diffusion properties, which means that the encryption of any plaintext bit in a given block is affected by every other plaintext bit in the same block. If a single bit is changed in plaintext, the ciphertext should change completely.

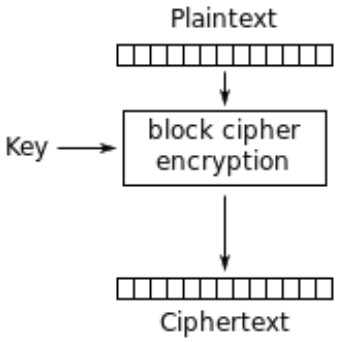


Figure 1-4: Block Cipher

In practice, the vast majority of block ciphers have a block length of either 64 bits such as DES (Data Encryption Standard) and 3DES (triple DES) or 128 such as the advanced encryption standard (AES).

1.4.1.2.1 Data Encryption Standard (DES)

The data encryption standard (DES) was the most widely used symmetric-key encryption algorithm for secure and authentic communications. It is a block cipher that takes 56-bit key and 64-bit plaintext message as inputs and outputs a 64-bit ciphertext.

As shown in Figure 1-5, DES consists of 16 identical rounds as well as an initial and final permutations. In each round, a different 48-bit round key derived from the primary 56-bit key is used. The round itself is composed from a Feistel Network as shown in Figure 1-6, which is a powerful structure and an important building block for many modern block ciphers including DES, the Soviet/Russian GOST as well as two more recent ones called Blowfish and Twofish ciphers.

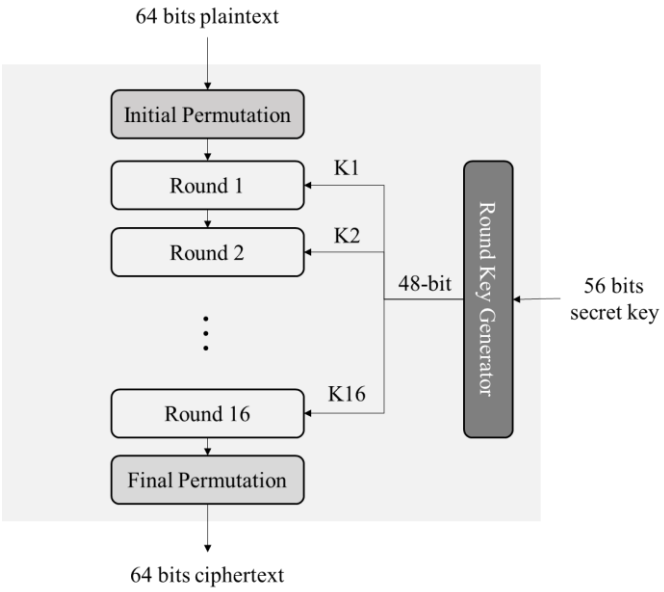


Figure 1-5: DES cipher architecture

A Feistel network takes two inputs, a data block and a sub-key, and output a block of the same size. First, the block is divided into two sub-blocks, right and left sub-block. The right block and the round key are fed into a pseudo-random function whose output is XORed with the left sub-block. The result of this operation will be the right sub-block of the network output. The left sub-block of the output is the exact copy of the right sub-block of the input without modifications.

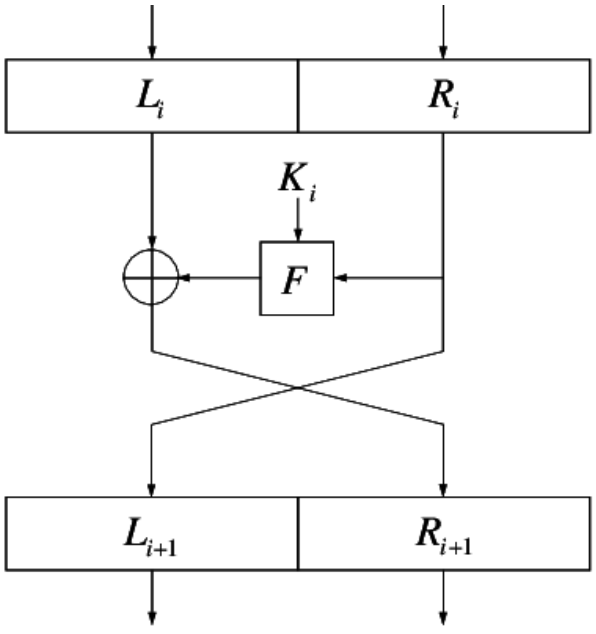


Figure 1-6: Feistel Network

An important advantage of Feistel networks compared to other cipher designs like substitution–permutation networks (SP Networks) is that the entire operation is guaranteed to be invertible regardless of the round function being invertible or not. Therefore, the round function can be made arbitrarily complicated, since it does not need to be designed with invertibility in mind.

Nowadays with the advancement in computing power, DES is considered insecure against brute-force attacks because of its small key. Therefore, DES should not be used in real applications anymore.

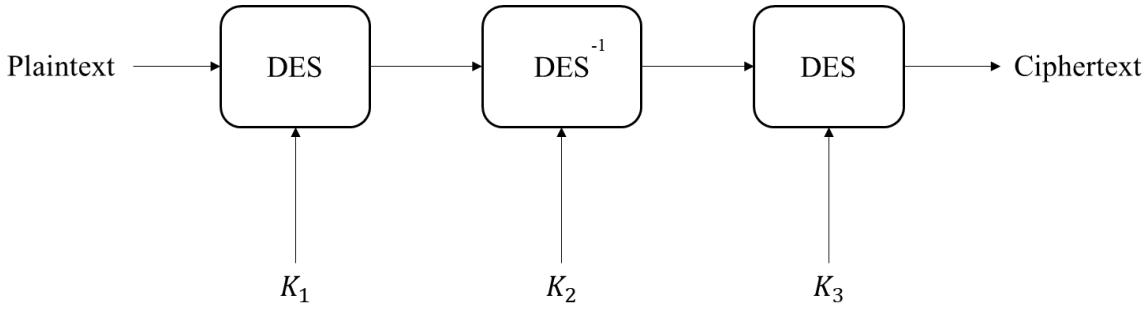


Figure 1-7: Triple-DES Cipher

One way of strengthening DES is using a derived algorithm called Triple-DES (3DES). It is an algorithm constructed of 3 stages of DES encryption and decryption algorithms using a different key for each stage as shown in Figure 1-7.

Another simple approach for strengthening DES is to use a method called key whitening. For this, two additional 64-bit keys k1 and k2 needs to be XORed to the plaintext and ciphertext, respectively, before and after the DES algorithm. According to equation Eq1 below.

$$DES_{k,k2,k3}(\mathrm{x}) = DES_k(x \oplus k1) \oplus k2 \qquad \textit{Eq 1}$$

1.4.1.2.2 Advanced Encryption Standard (AES)

In January 1997, the National Institute of Standards and Technology started an open selection process for a new encryption standard – the advanced encryption standard (AES). NIST encouraged cryptographers from all around the world to participate. The requirements were to support a block size of at least 128 bits, and three key sizes of 128, 192 and 256 bits. In November 2001, NIST selected the cipher named Rijndael developed by J. Daemen and V. Rijmen to be the AES.

Rijndael is a block cipher that supports different block and key sizes. Block and key sizes of 128, 160, 192, 224 and 256 bits can be combined independently. However, after being selected as the AES the block length was fixed to 128 bits and only key lengths of 128, 192 and 256 bits were chosen [9].

AES consists of the repeated application of a round transformation on the state. The number of rounds depends on the block length and the key length.

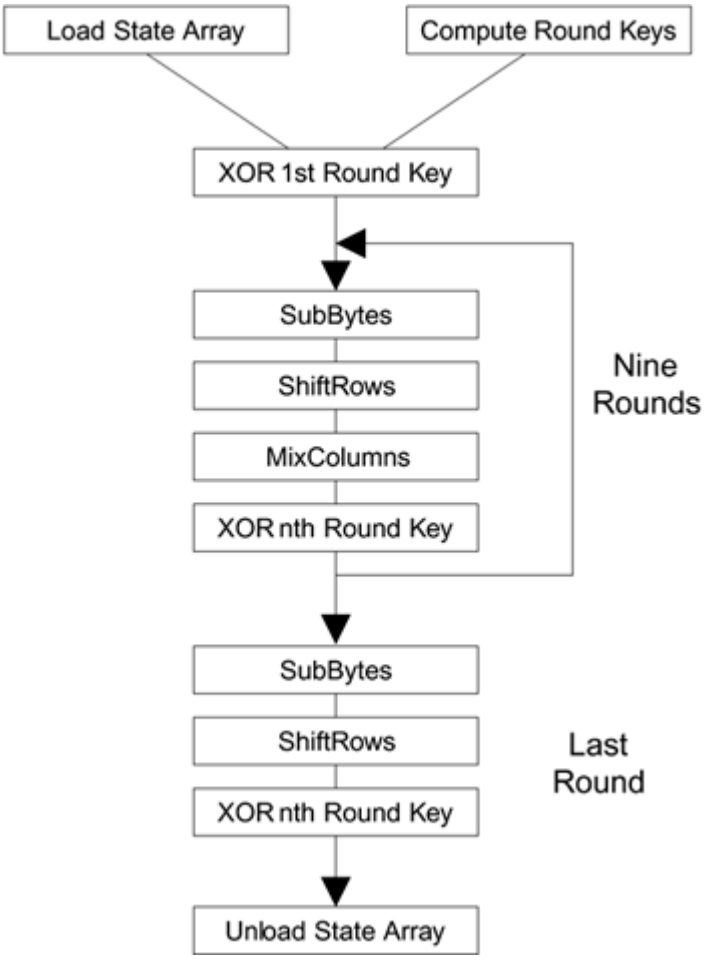


Figure 1-8: The Advanced Encryption Standard (AES)

AES accepts a block of data of 128 bits and a key of 128, 192 or 256 bits. The data block is first XORed with the first round key which is a sub-key derived from the provided key. Then 10 rounds are performed on the resulting block as shown in Figure 1-8. Each round consists of 4 steps except for the last one which has 3 steps only. The round steps are:

- **SubBytes:** a non-linear substitution where each byte is replaced with another according to a predefined S-Box.

- **ShiftRows:** a transposition step where the last three rows of the state are shifted cyclically a certain number of steps.
- **MixColumns:** a linear mixing operation which operates on the columns of the state, combining the four bytes in each column.
- **AddRoundKey**: an application of XOR operator between the state and the corresponding round key.

By design, the last round doesn't include the MixColumns step.

1.4.1.3 Modes of operation

Most of the time the data to encrypt is larger than one block. A mode of operation is the method used to encrypt large data using a symmetric block cipher. There exist a lot of modes of operation that differ in speed and level of security. If the length of the message is not a multiple of the block size, it must be padded to be a multiple of the block size prior to encryption.

1.4.1.3.1 ECB Mode

The Electronic Code Book (ECB) mode is the most straightforward way of encrypting a message. Given a block cipher with a block of size *b* bits, messages which exceed b bits are divided into b-bit blocks. As shown in Figure 1-9, in ECB mode each block is encrypted separately from the other blocks. There is no interaction between blocks or influence on each other. The block cipher by the way can be any block cipher algorithm like AES or 3DES.

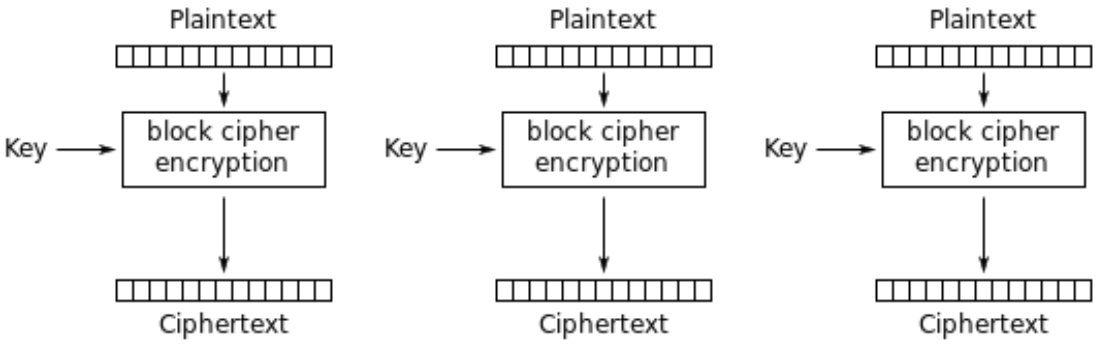


Figure 1-9: Encryption using ECB mode

This property of ECB mode of encrypting and outputting the result of each block separately makes it the most performant mode of operation since there is no other processing than the

encryption itself. However, this made ECB unsecure because it reveals too much patterns about the plaintext data.

Figure 1-10 represents the ECB penguin image, a famous experience where the pixels of a penguin image is encrypted using ECB mode. The result as shown below is an encrypted image that reveals a lot of patterns about the original image although every pixel has been encrypted.

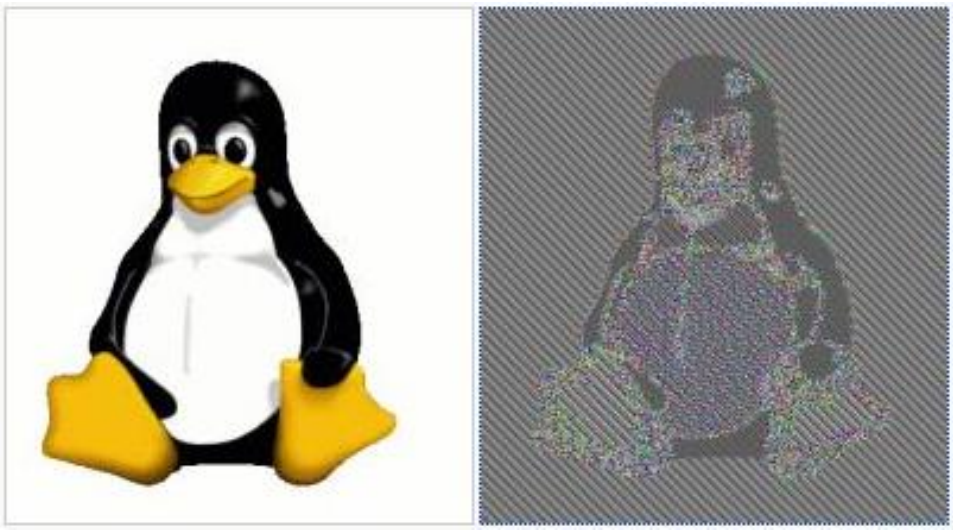


Figure 1-10: ECB penguin image

1.4.1.3.2 CBC Mode

Cipher Block Chaining (CBC) mode is a mode of operation that solved the security problem that ECB have. Unlike ECB where each block is encrypted separately and there is no influence of one block on the other, in CBC each ciphertext output depends on all ciphertext blocks that precedes it as well as a random initialization vector (IV) that should be provided upon each execution of the process. Therefore, even if two plaintexts are similar the ciphertext will be completely different because the initialization vector (IV) will not be the same.

CBC mode works as shown in Figure 1-11, in the first block the initialization vector bits are added to the plaintext block, the result is encrypted and output as the first ciphertext block. This ciphertext is then used as an initialization vector for the next block and so on. Each block in the ciphertext will depend on all the ciphertext blocks before it and on the initialization vector as well.

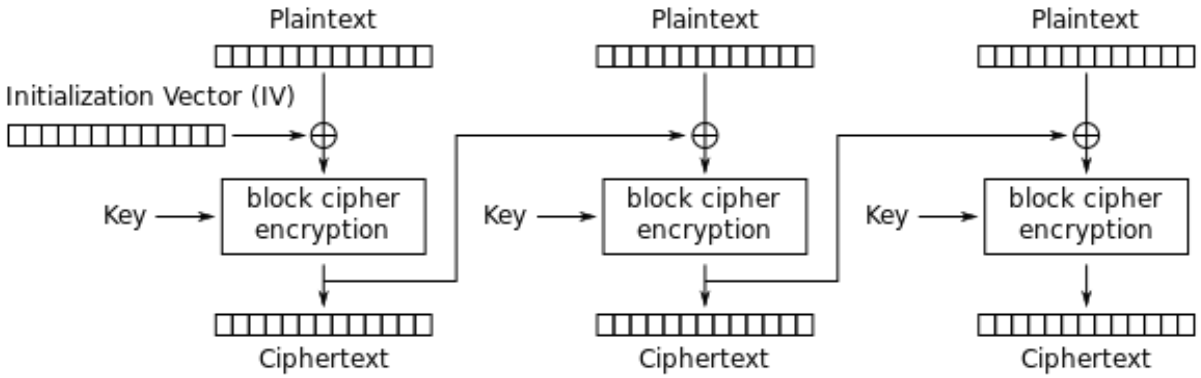


Figure 1-11: Encryption using CBC mode

Chaining cipher blocks solved the security problem that ECB had. However, this security comes at a performance cost. In CBC, the blocks' encryption cannot be parallelized and take advantage of modern hardware where CPUs have multiple cores and threads. In order to decrypt the last block, you need to decrypt all the blocks before it synchronously in the same thread.

1.4.1.3.3 OFB Mode

In the Output Feedback (OFB) mode a block cipher is used to build a stream cipher encryption scheme in a block-wise way.

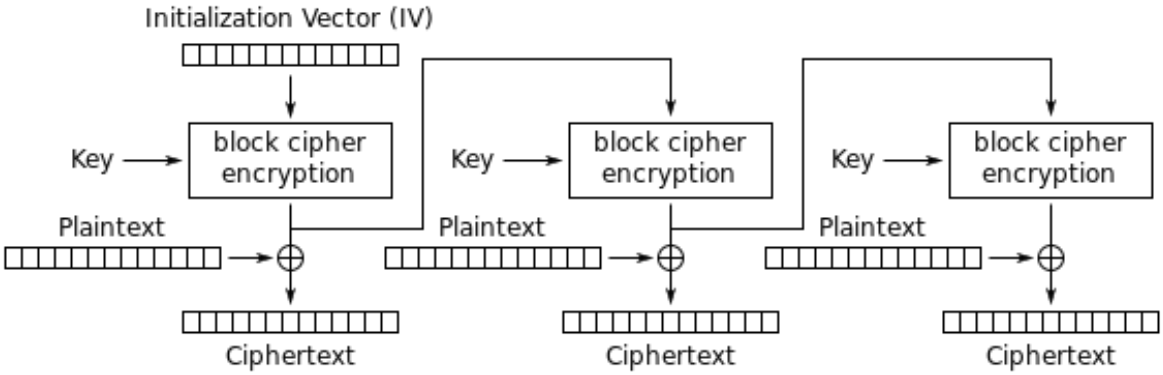


Figure 1-12: Encryption using OFB Mode

Figure 1-12 represents the encryption process in OFB mode. The process is quite similar to using a standard stream cipher because the key stream depends only on the Initialization vector

and doesn't depend neither on the plaintext or the ciphertext. First, the block cipher takes the initialization vector and encrypts it using the secret key, the resulting block is then fed into the block cipher and encrypted again and so on. This operation is repeated as many as the number of blocks contained in the plaintext. Those resulting blocks are then added to the plaintext (using XOR operation) block per block and the result will be the final ciphertext.

OFB mode uses only the encryption algorithm. In the decryption process, the same operations are repeated and the difference will be only in the last step where the XOR operation needs to be performed between the ciphertext and the encrypted blocks.

1.4.1.3.4 CFB Mode

The Cipher Feedback (CFB) mode also uses a block cipher as a building block for a stream cipher. It is very similar to the OFB mode mentioned earlier. However, the ciphertext is fed back instead of the output of the block cipher.

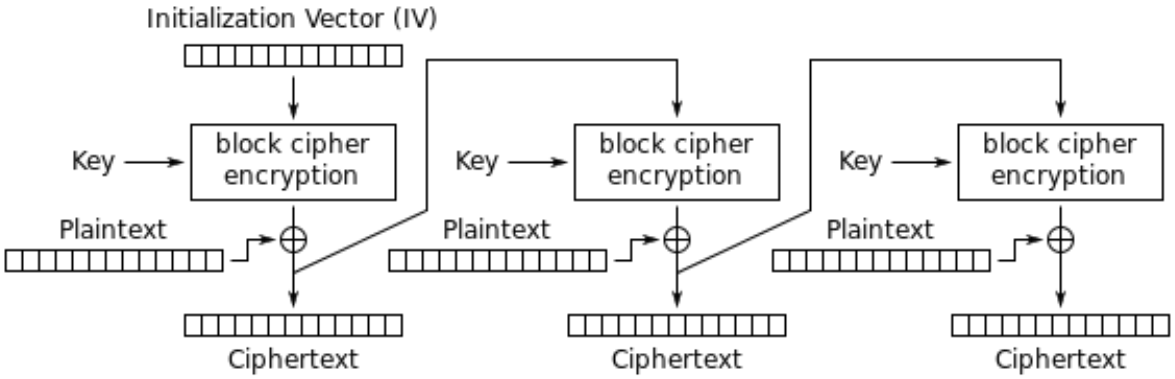


Figure 1-13: Encryption using CFB Mode

Figure 1-13 represents the encryption process in CFB mode. First, the block cipher takes the initialization vector and encrypts it using the secret key, the resulting block is then added to the plaintext block to get the first ciphertext block. Then this ciphertext block is used in place of the initialization vector in the next block and so on until the encryption of the whole plaintext is done.

1.4.1.3.5 Counter Mode

Counter mode is a mode that solved both security and performance problems of the modes above. CTR mode works by feeding an initialization vector into the block cipher. The output of the block cipher is then added to the plaintext and outputs the ciphertext. The initialization

vector should be different for every block to not add the same bit values to all blocks of plaintext as illustrated in Figure 1-14. Actually, counter mode uses a block cipher as a stream cipher.

The initialization vectors in CTR mode don't need to be random. The only condition they need to meet is uniqueness while the block cipher is using the same key. The IV is divided into two sets, the first set of bits is used as a Nonce (Number used Once) that should be unique as long as the key is fixed, while the other set is used as a counter that starts from 0 and increment by 1 after each block encryption.

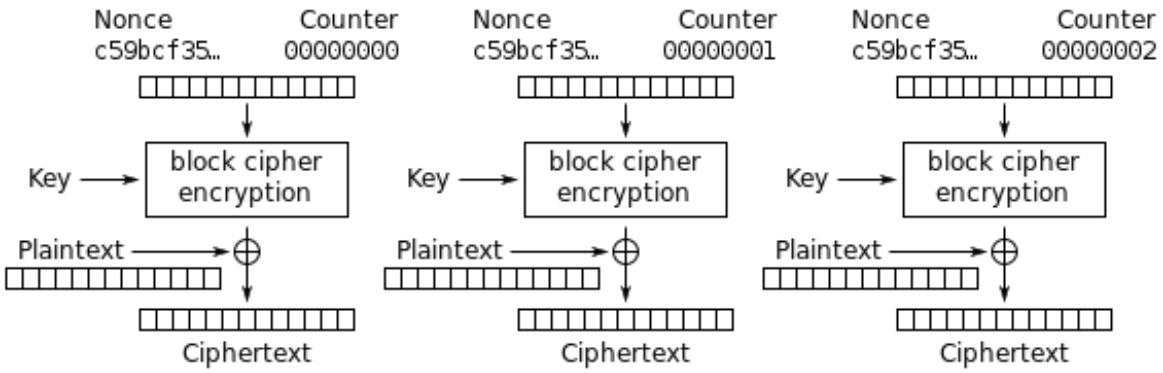


Figure 1-14: Encryption using Counter Mode

One attractive feature of the Counter mode is that it is highly parallelizable because it does not require any feedback from the previous blocks. The last block can be encrypted or decrypted without the need to encrypt or decrypt the previous blocks like in CBC mode. With counter mode, we can have two or more block ciphers running in parallel, increasing the speed proportionally. For applications with high throughput demands, e.g. in networks with data rates in the range of Gigabits per second, encryption modes that can be parallelized are very desirable [4].

1.4.1.3.6 Galois Counter Mode (GCM)

The Galois Counter Mode (GCM) is another mode of operation for symmetric-key ciphers. It is widely adopted for its security and performance. It is an authenticated encryption algorithm designed to provide both data integrity and confidentiality.

Figure 1-15 illustrates how GCM mode works. It works similar to Counter Mode by taking an initialization vector IV which combines a Nonce and a counter. It increments the counter on

each block and encrypt the new IV using the block cipher and XOR the result with the plaintext block. The encryption output is then fed into a MAC (Message Authentication Code) generation algorithm to compute an authentication tag of the ciphertext. The final output of GCM that will be sent to the recipient will contain the IV, the ciphertext and the authentication tag. This way, the recipient is the only one who can decrypt the message and can make sure that the message was created by the real sender and that nobody tampered with the ciphertext during transmission. GCM provides confidentiality, authentication and integrity and it is widely used in various applications.

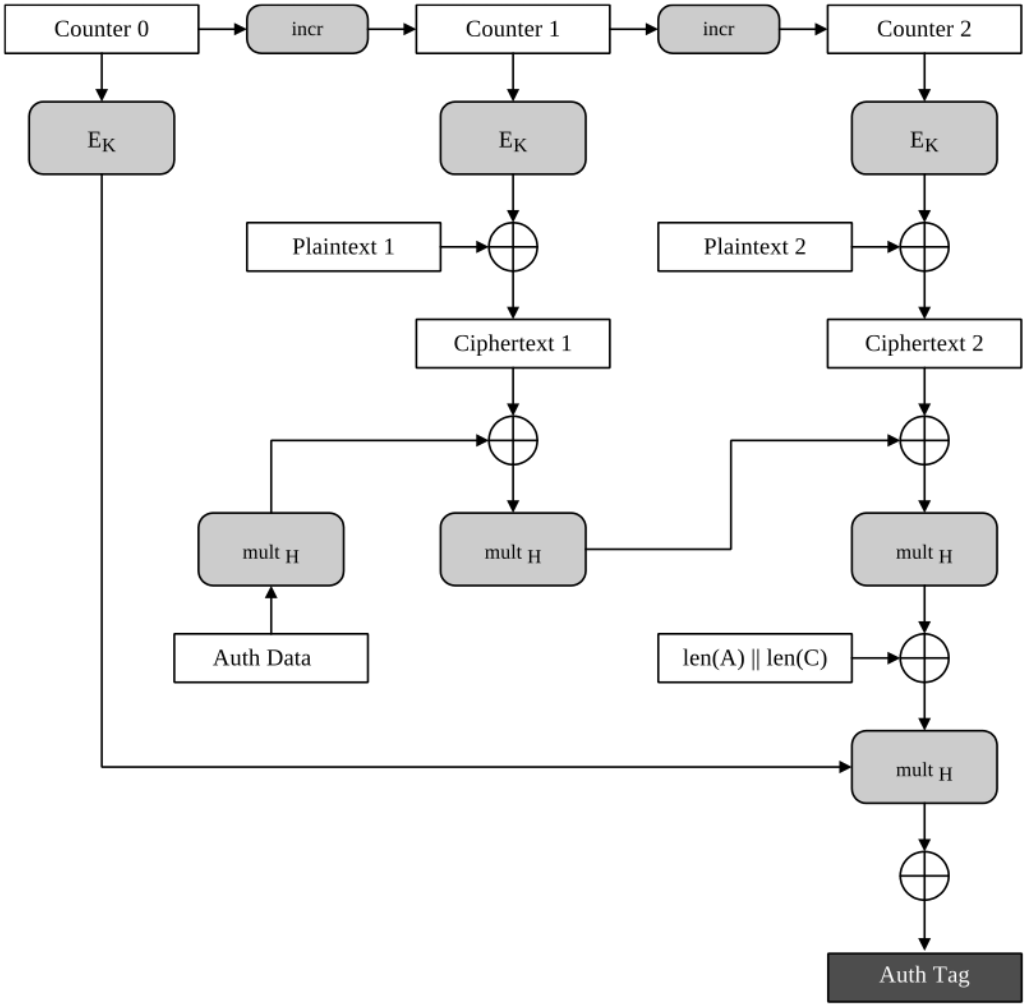


Figure 1-15: Encryption using Galois Counter Mode (GCM)

1.4.2 Asymmetric ciphers

Asymmetric-key or also known as public-key cryptography is a cryptographic system that involves two algorithms: encryption and decryption but involves two different keys. One of the

keys is used for encryption called a public key and the other one is used for decryption known as a private key. While the public key is publicly known to everyone, the private key is only known the intended recipient of the cipher data [10].

The relationship that exists between the keys and algorithms is that any data encrypted using the public key should be decrypted using the private key only and vice-versa. The private and public keys used in the process are generated using a key generator which generate a valid pair of keys each time it runs [7].

Asymmetric ciphers are known to be slow compared to symmetric ciphers and to offer the same level of security they should use keys larger in size. However, one of the most used applications of them is shared secret key establishment over a public channel. RSA, and EIgamal are some examples of well-known Asymmetric-key algorithms [4].

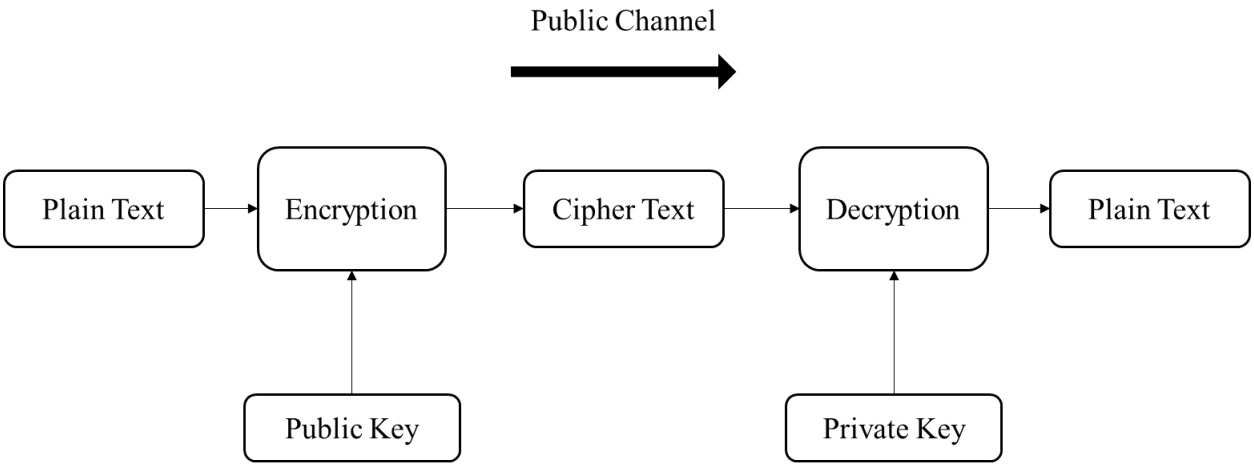


Figure 1-16: Public key cryptography principle

The flow of communication using an asymmetric ciphers goes like illustrated in Figure 1-16. The sender asks for the receiver's public key to encrypt his message. He then sends the encrypted message to the recipient. The recipient receives the ciphertext and decrypt it using his private key.

1.4.2.1 The RSA Cipher

RSA cipher is one of the most famous and widely deployed public-key cryptosystems. It was published in 1978 by three cryptographers: Ron Rivest, Adi Shamir and Leonard Adleman [6]. The name RSA is an acronym comes from their last names' first letters.

The security of RSA relies on the well-known mathematical difficulty of factoring the product of two large prime numbers named the "factoring problem". Until now, RSA is considered safe and there are no published methods to defeat the system if a large enough key is used [1].

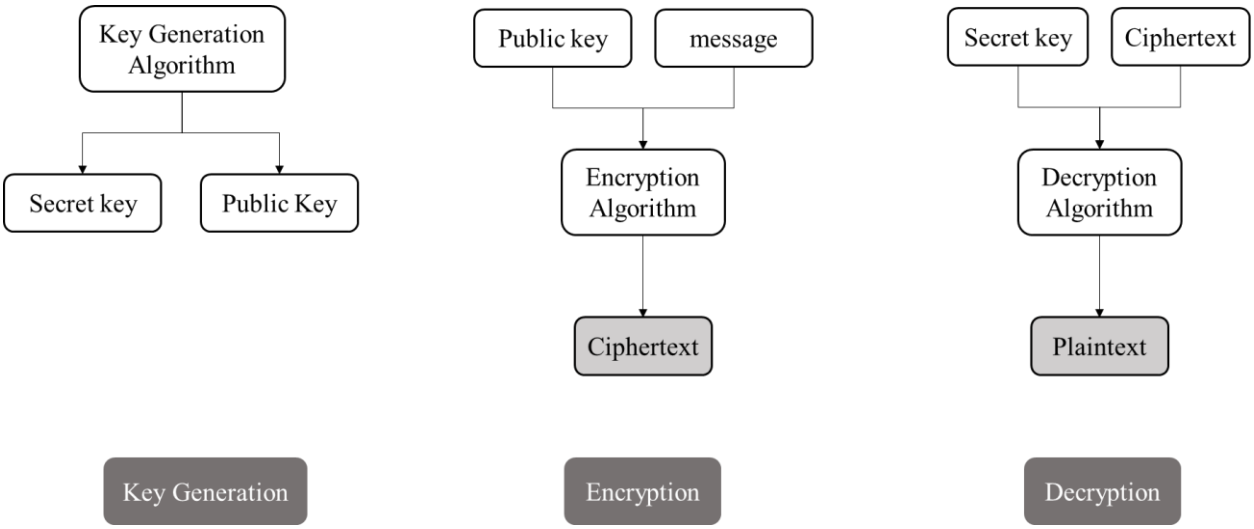


Figure 1-17: RSA Cryptosystem

RSA is composed of 3 algorithms as illustrated in Figure 1-17: Key Generation, encryption and decryption algorithms. The key generation algorithm generates a pair of private and public keys. The encryption algorithm takes the message and the public key as inputs and outputs a ciphertext. To decrypt that ciphertext it needs to be fed into the decryption algorithm with the private key [4].

The RSA key generation algorithm has 3 steps:

1. Choose 2 large distinct primes p and q, and compute $n = p \times q$.
2. Choose e that is prime to $\varphi(n)$. The pair (n, e) is published as the public key.
3. Compute d with $ed \equiv 1 \, mod \, \varphi(n)$. The pair (n, d) is used as the secret key.

$\varphi(n)$ is called *Euler phi function*, which is the number of integers in the range $[1, n-1]$ which are prime to n. When n is the product of 2 primes, there is a simple formula to calculate $\varphi(n)$ as in equation eq2

$$\varphi(n) = (p - 1)(q - 1) \qquad \textit{Eq 2}$$

Once the private and public keys are generated. The encryption is then performed by rising the message to the power of the public key as defined by equation 3 below.

$$c = m^e \qquad \textit{Eq 3}$$

The decryption process on the other hand is similar to the encryption. We got the message back by rising the ciphertext to the power of the private key as shown in eq4.

$$m = \mathrm{c}^d \qquad \textit{Eq 4}$$

The security of RSA relies on the difficulty of finding the composite $\boldsymbol{n}$ factorization. If an adversary successfully gets the factors $\boldsymbol{p}$ and $\boldsymbol{q}$ of $\boldsymbol{n}$, he will also know $\varphi(n) = (p - 1)(q - 1)$, and then he can derives $\boldsymbol{d}$ from the public encryption key $\boldsymbol{e}$ using an algorithm named extended Euclidean algorithm.

1.4.3 Hash Functions

Cryptographic hash functions are one of the most important cryptographic primitives. They are one way algorithms that take an arbitrarily sized block of data and encrypt it into a fixed-sized bit string called the cryptographic hash value or message digest as shown in Figure 1-18. It can be looked at as a message unique representation or fingerprint [4]. If a single bit changes in the data, the hash function will output a completely different hash value. Unlike other cryptographic algorithms seen before, hash functions do not have a key.

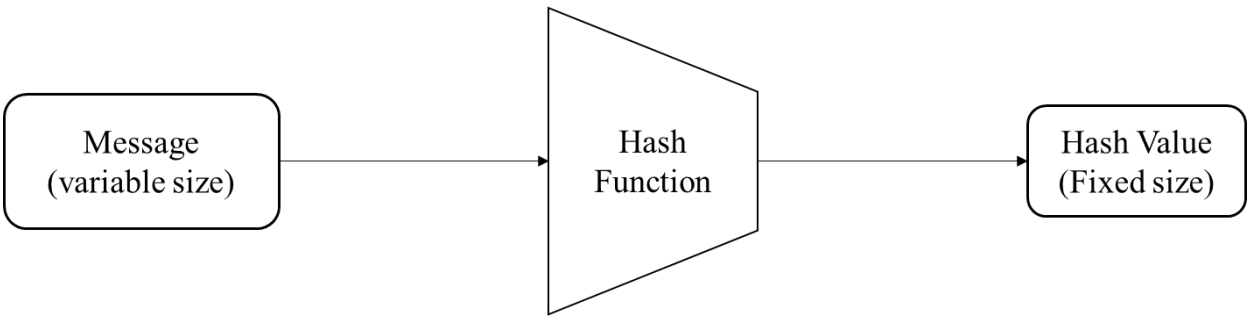


Figure 1-18: Hash functions work principle diagram

There are some requirements for an algorithm to meet in order to be considered as a good hash function. The first requirement is that the hash function should be one way. Which means, given a hash value $h(x)$ of a message x, it should be computationally infeasible to calculate the message x back. The other requirement is that a hash function should be collision resistant. This means, given a message $x1$ or its hash value $h(x1)$, it should be computationally infeasible to find a message $x2 \; ! = \; x1$ but has the same hash value $h(x1) \; = \; h(x2)$.

Hash functions play an important role in various applications such as data integrity, storing passwords in databases, deriving cryptographic keys from users' passwords …etc.

1.4.4 Message Authentication Codes

Message Authentication Code (MAC) is a tag attached to a message to ensure the integrity and authenticity of the message. MACs share some similarities with cryptographic hash functions, however, they address different security requirements. The purpose of a MAC is to authenticate the source of a message and its integrity [10]. Unlike a cryptographic hash, the MAC can be generated only by the sender or the intended recipient who has access to the secret key.

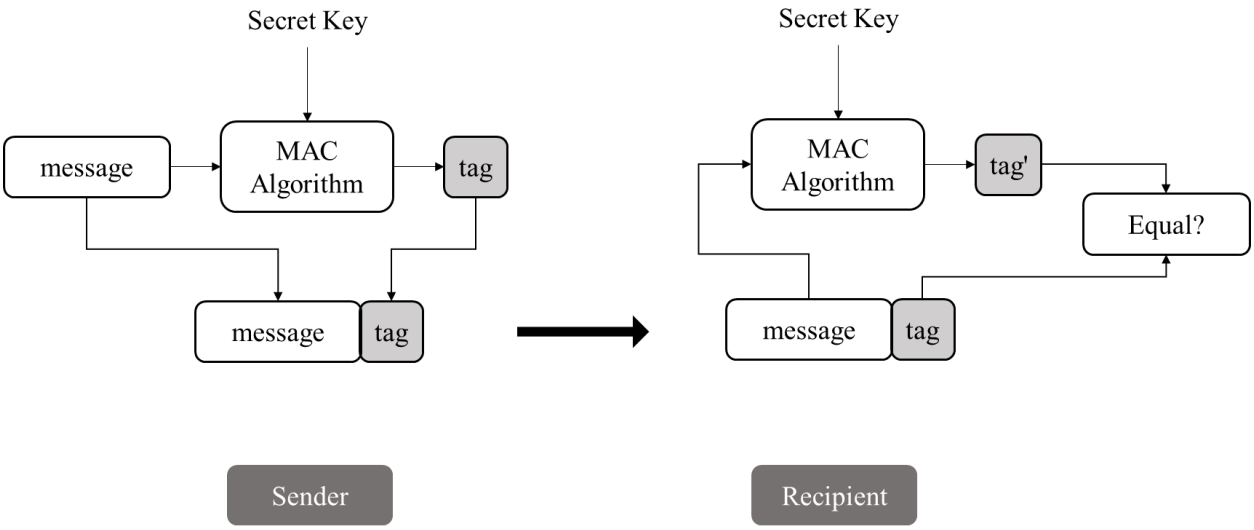


Figure 1-19: Message Authentication Code (MAC)

With the use of MACs, no message manipulation can happen without being detected. The MAC is calculated as a function of the message and the shared secret key to get a tag of a fixed length. Then, both the message and the authentication tag get sent to the recipient. When receiving the message and the tag, the recipient verifies the message integrity by recalculating the tag using the shared secret key as illustrated in Figure 1-19. If the tag he finds corresponds to the tag he received the message is valid. Otherwise, the message has been manipulated intentionally or modified accidentally by a third party.

1.4.5 Digital Signatures

Digital signatures are important cryptographic tools that are widely used today. They are built on top of public-key cryptography and their applications range from digital certificates for

secure e-commerce to legal signing of contracts to secure software updates. The purpose of digital signatures is to provide authentication and non-repudiation [2].

Digital signatures share some similarities with MACs since they both verify integrity. However, they differ in the non-repudiation assurance [7]. MACs can protect the message from third parties. However, in case of a conflict or a disagreement between the two ends of the encryption, it is not possible to prove in front of a neutral party who encrypted the message.

In order to see how the above mentioned limitation of MACs can be a problem, lets imaging a scenario where Alice is a car dealer and Bob is a customer. Both Alice and Bob agreed on a shared secret key to communicate securely with each other. Bob ordered a specific car from Alice and by sending here an encrypted order with the shared secret key and he was aware of the non-return policy of Alice. After few days, bob changed his mind and tell Alice that he is not going to pay for the car and he no longer wants it. Then, Alice decided to take legal action against him. She showed the encrypted order that Bob sends to her but Bob denied that the order was from his part and he claimed that the order was written by Alice herself. In this case the judge decided that he cannot be sure if Bob was really the sender or Alice. Thus, Bob probably gets away with his dishonesty. However, digital signatures can prevent that from happening.

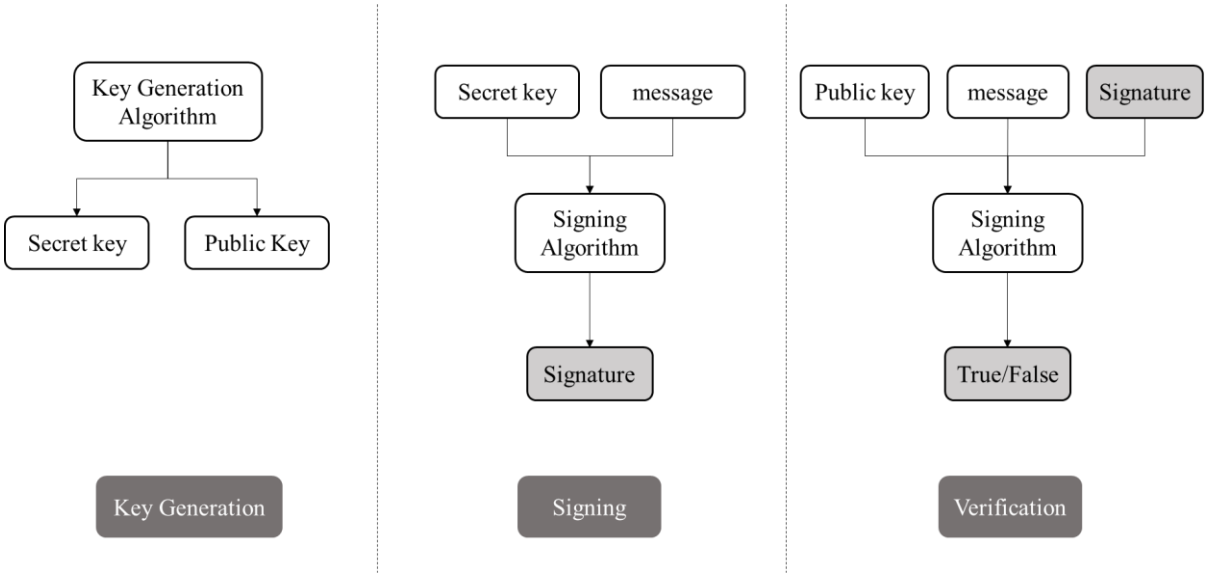


Figure 1-20: Digital Signature system components

A digital signature system usually consists of three algorithms as shown in Figure 1-20:

- A key generation algorithm that selects a private key uniformly at random from the key space. The algorithm outputs the private key and a corresponding public key.
- A signing algorithm that produces a signature given a message and a private key.
- A signature verifying algorithm that, given the message, the public key and signature, either accepts or rejects the message's claim to authenticity and returns either “true” or “false”.

1.4.6 Key Exchange Protocols

Before key exchange protocols were invented, keys exchange between two parties who want to communicate securely was done through a private channel. There was no mean that allow agreement on a shared secret key in public without the key being known by third parties. Key Exchange Protocols were invented to solve this problem. Their invention made it possible for two parties who want to communicate securely, and who have never met before, to agree on a shared secret key in public without the secret being known by third parties.

1.4.6.1 Diffie-Hellman Key Agreement

Diffie-Hellman key exchange protocol provided the first practical solution to the key distribution problem. In 1976, W. Diffie and M.E. Hellman published their fundamental technique of key exchange together with the idea of public-key cryptography in the famous paper, “New Directions in Cryptography” [5]. Exponential key exchange enables two parties that have never communicated before to establish a mutual secret key by exchanging messages over a public channel. This technique was a revolutionary concept that opened the door for public key cryptography.

Figure 1-21 illustrates how Diffie-Hellman protocol works. Let $\boldsymbol{p}$ be a sufficiently large prime (1024 bits or more). Let $\boldsymbol{g}$ be a primitive root in Z_p^*. $\boldsymbol{p}$ and $\boldsymbol{g}$ are publicly known. Alice and Bob can agree on a secret shared key by executing the following protocol:

1. Alice chooses a, $0 \leq a \leq p - 1$, at random, sets $A = g^a \bmod p$, and sends the result A to Bob.
2. Bob chooses b, $0 \leq b \leq p - 1$, at random, sets $B = g^b \bmod p$, and sends the result B to Alice.

3. Alice computes the shared key $k = B^a = (g^b)^a$

4. Bob computes the shared key $k = A^b = (g^a)^b$

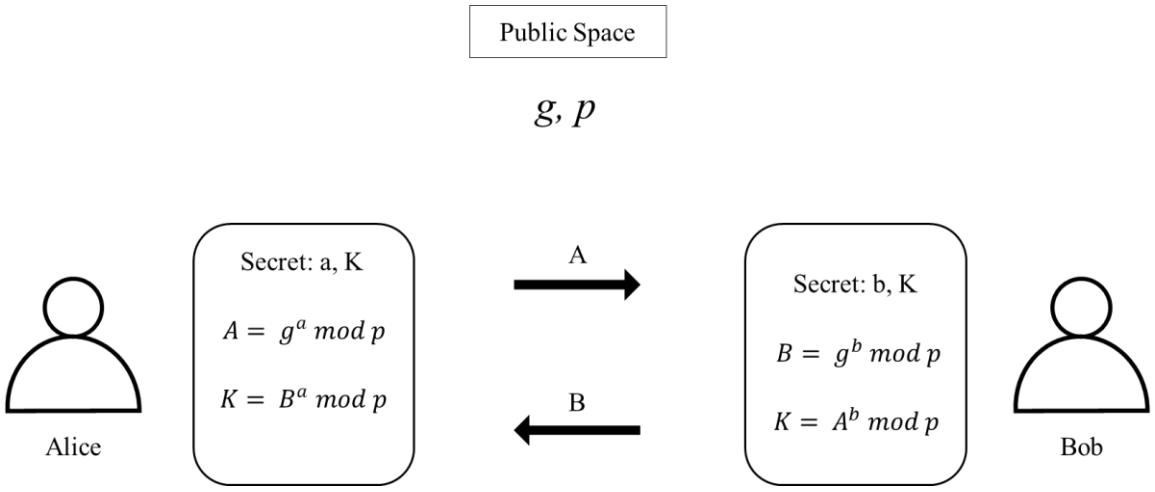


Figure 1-21: Diffie-Hellman key exchange protocol

Thus both Alice and Bob end up with the same shared secret key ***k*** that is known to them only. If an attacker wants to know the secret that Alice or Bob choose. He needs to solve the Discrete Logarithm Problem. However, this is a hard mathematical problem to solve.

1.5 Cryptanalysis

Cryptography is the art and science whose goal is to secure data and communications from eavesdropping and tampering by the non-authorized persons or systems. Cryptanalysis on the other hand is the opposite. It is the art and the science of deciphering or forging communications that was secured by cryptography. Cryptography and cryptanalysis are two competing fields. The success of one of them in a given area means the failure of the other [4].

Since the early days of cryptography, cryptanalysts were working hard on breaking the encryption system and retrieve data. Over time, a lot of cryptanalysis methods had been developed and added to the toolset of cryptanalysts.

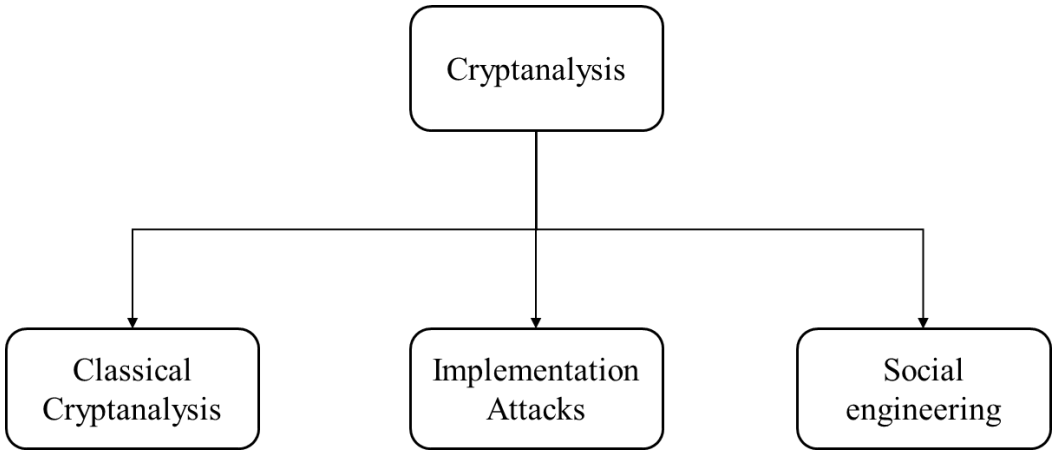


Figure 1-22: Overview of cryptanalysis

Cryptanalysis nowadays can be divided into 3 main categories as illustrated in Figure 1-22: Classical Cryptanalysis, Implementation attacks and Social Engineering methods [4].

1.5.1 Classical Cryptanalysis

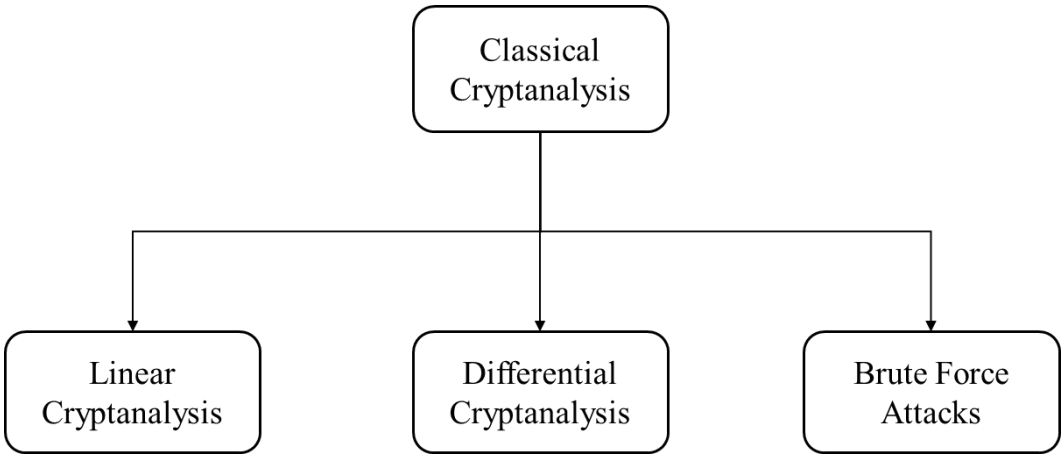


Figure 1-23: Overview of Classical Cryptanalysis

Classical cryptanalysis is the science of recovering the plaintext or the encryption key from the ciphertext. It focuses on studying the internal structure and underlying mathematical concepts of the system and try to find vulnerabilities. If any weaknesses are found, they will be used to get information about the key or the plaintext which is considered a failure of the encryption system. As shown in Figure 1-23, differential cryptanalysis, linear cryptanalysis and brute-force attacks are well-known examples of this category.

1.5.1.1 Differential cryptanalysis

Differential cryptanalysis is performed by looking at ciphertext pairs, where the plaintext has a particular difference known to the attacker. The exclusive-or of such pairs is called a differential and certain differentials have certain probabilities associated to them, depending on what the key is. By analysing the probabilities of the differentials computed in a chosen plaintext attack some information about the underlying structure of the key is revealed sometimes [7].

1.5.1.2 Linear Cryptanalysis

A good cipher should always contain non-linear components. However, linearity could be introduced by cipher designer unintentionally. The idea behind linear cryptanalysis is to approximate the behaviour of the non-linear components with linear functions. Some probabilistic analysis is then performed to determine information about the key [7].

1.5.1.3 Brute force attacks

Brute force attacks are the simplest form of attacks that can be conducted against any type of ciphers using trial and error. They consist on performing an exhaustive search for the cipher encryption key trying all possible values. The longer the key the more resistant is the cipher against this kind of attacks.

Data Encryption Standard (DES) uses a key of 56 bits, which was the primary reason for its cryptanalysis after few decades of its invention. Nowadays 56 bits key are crack-able in few hours or days as illustrated in Table 1 [4].

Table 1: Brute-force attacks estimated time on symmetric ciphers with by key lengths

Key Length	**Security Estimation**
< 64 bits	few hours or days
112 -128 bits	several decades in the absence of quantum computer
256 bits	several decades, even with quantum computers that run the currently known quantum computing algorithms

Nowadays, key lengths of 128 bits or more are encouraged to resist brute-force attacks. However, with the development of quantum computers this key size will fall in the scope of

exhaustive search too. That's why researchers and governments who care about the security of their data should keep an eye upon quantum computers' development. However, a long key will only help if the cipher structure is safe. If any analytical attack is possible, key size will be less relevant.

1.5.2 Implementation Attacks

Implementation attacks are a relatively new type of attacks. They have become popular with the rise of devices and integrated circuits that perform the cryptographic operations. This type of attacks doesn't attack the abstract cryptographic algorithm but their practical implementations in cryptographic devices [11]. These devices are components in a system that either perform the actual cryptographic operations or handle the data needed for these operations. Examples include dedicated cryptographic hardware modules like encryption coprocessors, and memory circuits that store secret keys or even general purpose hardware like microcontrollers that execute software implementations of cryptographic algorithms. Cryptographic devices tend to be closed platforms in order to increase the protection from Attacks and they come in many shapes such as smart cards or USB tokens.

Implementation attacks are relevant against cryptosystems to which an attacker has physical access. In most Internet-based attacks against remote systems, implementation attacks are usually not a concern [4]

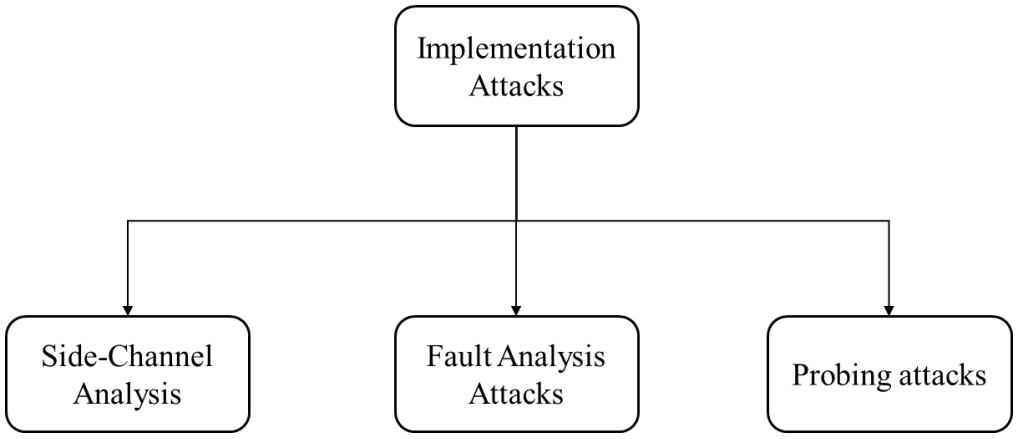


Figure 1-24: Overview of Implementation Attacks

Implementation attacks can be classified into 3 sub-categories as shown in Figure 1-24: Side-channel attacks, Fault attacks and Probing attacks.

1.5.2.1 Side-Channel Analysis

Side-channel analysis (SCA) attacks have received a lot of attention in the last decade. They are passive attacks which exploit physical signals emitted by normally operated cryptographic devices [11]. The most common side channels are execution time, power consumption, temperature and electromagnetic radiation. The basic principle of SCA attacks is to determine secret keys or part of them from their influence on the side-channel signals.

1.5.2.2 Fault analysis attacks

Fault analysis attacks are a general more complex type of attacks than SCA attacks because they are active attacks that require tampering with the cryptographic system [11]. Their principle is to induce faults in a cryptographic circuit and make it behaves abnormally or delivers incorrect results. The delivered incorrect result reveals information about the secret key.

1.5.2.3 Probing attacks

Probing attacks are another passive but yet powerful type of implementation attacks. In this type of attacks, the inputs and outputs of internal logic cells and blocks are electrically connected to a monitoring device to read out their state while the cryptographic circuit operates normally. This type of attacks is getting harder and more complicated with the constantly shrinking dimensions of cryptographic devices. However, the effort may still be worth it because successfully planted probes in the right locations allow in the best case to directly read out bits of secret keys.

1.5.3 Social Engineering Attacks

Social engineering is a mean of breaking security and bypass data encryption, not through cipher systems analysis but through the psychological manipulation of people into performing actions or revealing private information. Examples of social engineering includes the act of behaving as an authority, intimidation and creating a state of scarcity and urgency in order to lead people to disclose their confidential information.

An attacker always looks for the weakest link in a cryptosystem either the weakness is in the system itself or in the people who use it. That means that in order to keep systems safe, a strong algorithm should be chosen and an attentive eye should be kept on making social engineering and implementation attacks not practical.

1.6 Conclusion

In this chapter, we gave a brief introduction to the field of cryptography. We saw some history about cryptography and how that science had started. The primary techniques and tools of the field as well as their categories and applications were then presented. Finally, we looked at the techniques of breaking security. We will exploit some of those techniques in chapter 3 to break a recently published image encryption scheme.

Chapter 2

Overview on Image Encryption

Index

Chapter 2

Overview on Image Encryption

2.1 Introduction

New problems of security and privacy have emerged with the presence of modern communication systems. Images nowadays are one of the most usable forms of information. Image and video encryption is applied in various fields, including multimedia systems, medical imaging and military communications.

Due to the importance of images in the modern world, the study and development of encryption techniques specific to images have known a lot of interest recently. Many algorithms have been proposed and many techniques have been developed by security researchers.

In this chapter, some basics about digital images are given. Then, different images encryption techniques and their evaluation metrics are presented. This chapter will give the reader the necessary notions to understand the following chapters.

2.2 Overview on Digital Images

An image is a visual representation of something fed into the visual system to carry information. However, the term is usually used when referring to the static representation of scenes or objects either on paper or on screens. Images can be classified based on their storage medium into two categories: analog and digital images. Digital images are simply the type of images that are viewed on monitors of electronic devices like phones and computers. Analog images on the other hand are the images found elsewhere on films, paper, T-shirts …etc. Digital images will be the focus of our study in this thesis.

2.2.1 Digital Images

A digital image is an electronic representation of a visual stored in binary format on an electronic storage medium. The common representation model of a digital image is the pixels' matrix model.

2.2.1.1 Grayscale Images

A grayscale image is the simplest form of digital image. It is usually represented as a matrix of *M* * *N* elements. Each element of the matrix is called a pixel. The pixel is the smallest building block of a digital image. It is encoded in 8 bits of data and takes values in the range of [0, 255] of intensity or so called gray value. A pixel with value equal to 0 is a totally black pixel. As the intensity increases the pixel becomes brighter until it reaches 255 which is a totally white pixel.

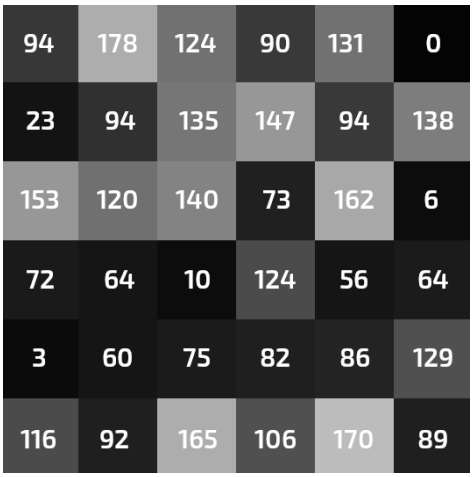


Figure 2-1: 6 by 6 Grayscale Image

Figure 2-1 represents a 6 * 6 grayscale image. It has 6 rows and 6 columns, that means a total of 36 pixels. The gray value of each pixel is written on inside the pixel. As noted, the bigger the value the brighter is the pixel.

2.2.1.2 The Color Image

The color image on the other hand is simply the superposition of 3 grayscale images called channels which have the same dimensions. Each channel represents the pixel intensities of one of the primary colors (Red, Green and Blue). Those three colors are used to represent all other colors. Given the right values, graphics cards know how to display the right corresponding color on digital screens.

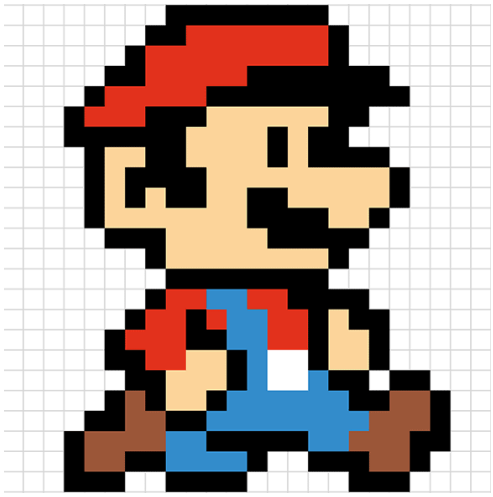


Figure 2-2: A 24 by 24 Color Image

Figure 2-2 represents a 24 by 24 color image of the famous video-game character Mario. Under the hood this image is composed from 3 channels corresponding to the RGB colors as illustrated in Figure 2-3 below.

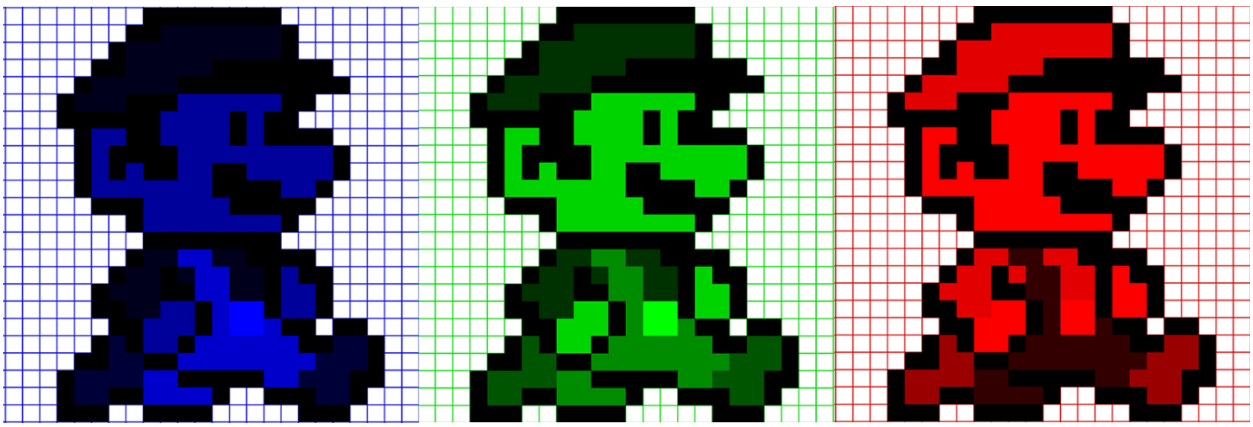


Figure 2-3: Decomposition of Mario image into RBG channels

2.2.2 Basic Operations on Images

In digital image processing, there exist many operations that can be performed on images. The basic operations that are going to be used in the following chapters are described below.

2.2.2.1 Pixel-wise arithmetic

Some of the most common operations performed on digital images are arithmetic operations like addition and multiplication (subtraction and division are addition and multiplication respectively). Those operations can be applied either between two images or between an image and a scalar value. Adding a scalar value to an image means increasing the value of all pixels

of that image by that value. On the other side, adding two images together means adding pixels having similar positions together.

When doing arithmetic operations on images and the value of a given pixel passes 255 ascending or 0 descending, two methods are used to deal with that, thresholding or modulo. Thresholding is fixing the upper threshold to 255 and the lower threshold to 0 if values goes above 255 or below 0. The second method is the use of modulo operator. In this case all arithmetic operations are calculated with modulo 255.

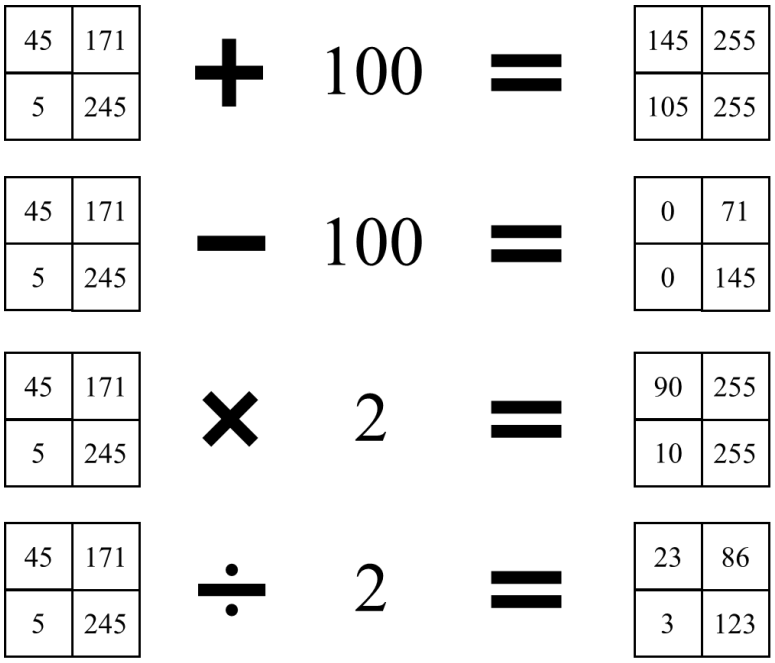


Figure 2-4: Image arithmetic operations with thresholding

Figure 2-4 illustrates how arithmetic operations are performed on images with thresholding. As seen in this example, values bigger than 255 and smaller than 0 are fixed to 255 and 0 respectively. In case when the pixel value is a non-integer as a result of division, the value is rounded to the nearest integer.

Figure 2-5 illustrates image arithmetic operations with a scalar with modulo. Values above 255 and below 0 are calculated modulo 255. The modulo needs to be positive always because pixels values are integers in the range from 0 to 255.

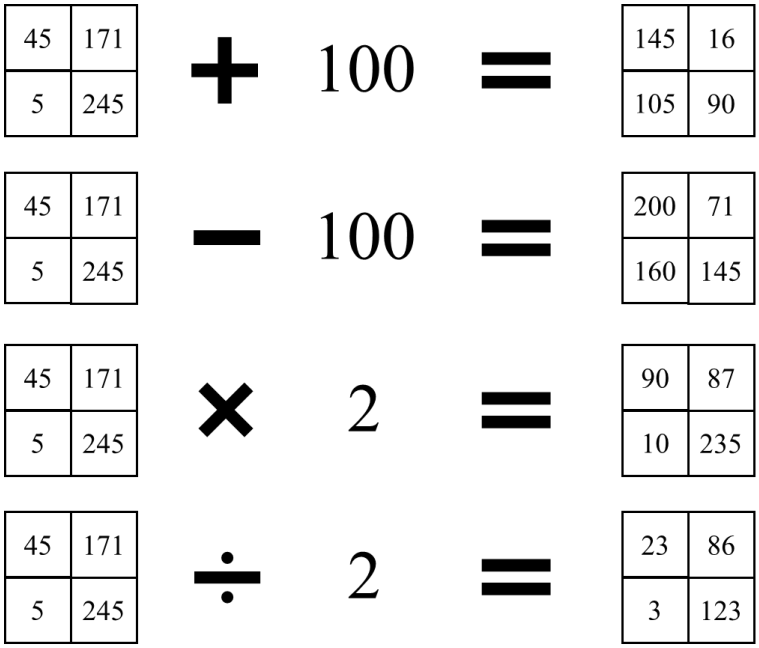


Figure 2-5: Image arithmetic operations with modulo

The choice between those two techniques of dealing with out-ranged values depends on the specifications of each application.

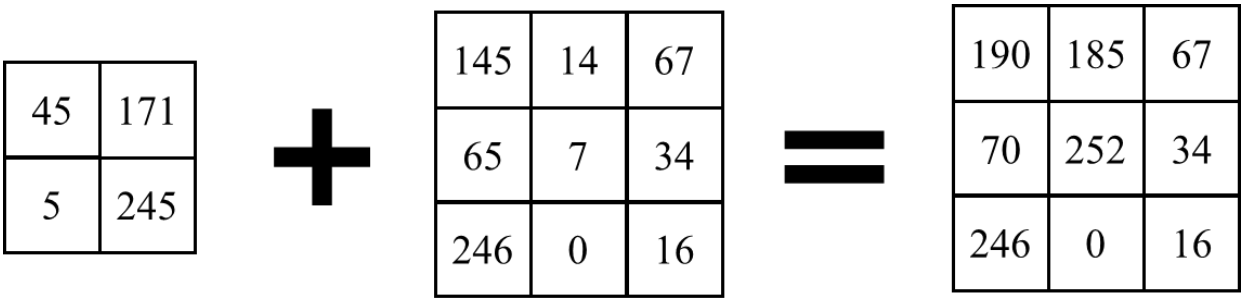


Figure 2-6: The addition of two images

Figure 2-6 shows how two images are added together. The addition is done pixel-wise. Each two pixels having similar positions are added together. If images are of different dimensions, the smaller image is padded with zeros.

2.2.2.2 Pixel-wise Convolution

convolution is a mathematical operation on two functions (f and g) that produces a third function *(f * g)* that expresses how the shape of one is modified by the other. It is the integral of the product of the two functions after one is reflected about the y-axis and shifted over the other.

In image processing, convolution is the process of adding each element of the image to its local neighbors, weighted by a small matrix called a kernel. Kernel convolution is used for different purposes like: blurring, sharpening, edge detection, and more.

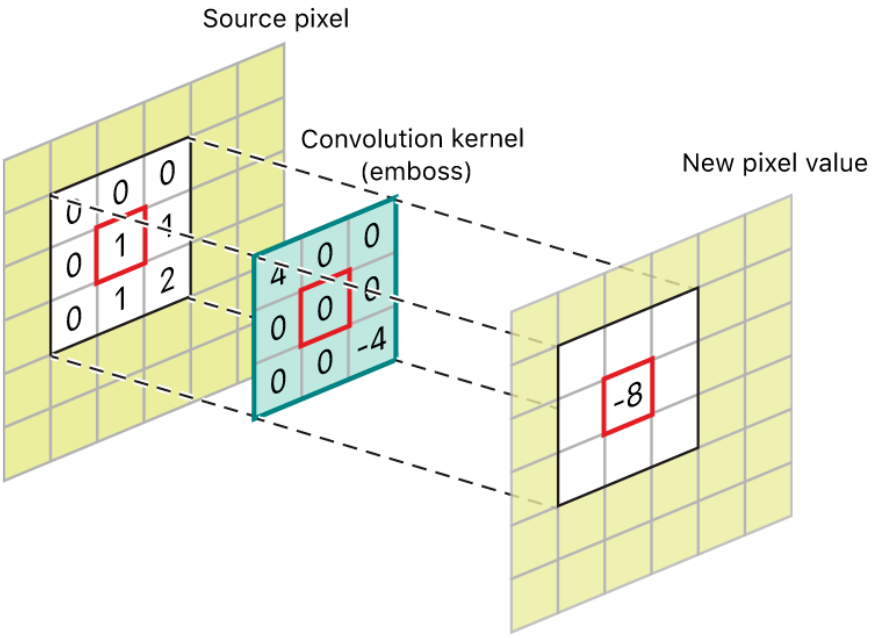


Figure 2-7: Image Convolution

Figure 2-7 illustrates how convolution is calculated using an embossing kernel as an example. The center element of the kernel is placed over the source pixel. The source pixel is then replaced with a weighted sum of itself and the nearby pixels

2.2.2.3 Pixel-wise Permutation

Pixels permutation is a fundamental operation in digital image processing. It is used heavily in image encryption. It is the change of pixels' positions with each other using a predefined map or permutation vector.

Figure 2-8, illustrates how image permutation is performed using a permutation vector. First, the image is reshaped into a single row. Then the permutation is done using the permutation vector by moving the pixel to the index corresponding to the permutation vector element value. For example, the first pixel is moved to the 4^{th} position, the second pixel to the 1^{st} and so on. After the permutation is done, the image is reshaped back to its original shape.

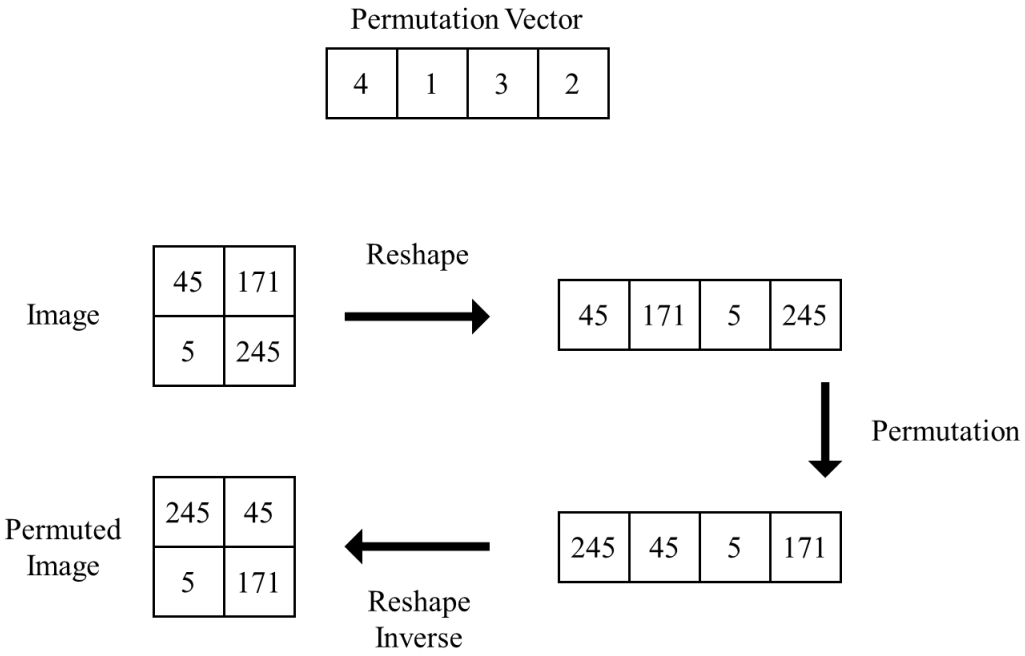


Figure 2-8: Image Permutation

Permutation is commonly used in image encryption schemes to break the correlation between adjacent pixels and scramble the image. Alongside with pixel substitution, they form the building blocks of many modern image cryptosystems [12].

2.2.3 Common Image Formats

When it comes to storing images on disk. Images are stored in many formats. Each format has advantages and disadvantages. Images can be classified into 2 categories: vector images and raster images [13].

Vector images are mathematical formulas that establish points on a grid to build images. They are great for images that need to be regularly resized because they can adjust in size without losing resolution.

Raster images on the other hand are comprised of a fixed number of colour pixels. This means if the file is resized, its resolution could be compromised. This type represents the majority of images found online and in print. Raster images are the type of images that we care about in this thesis. Following are some of the most common file formats.

2.2.3.1 BMP Format

BMP is short for bitmap. It is a raster image format that maintains high image quality and a large amount of details while treating each individual image pixel as its own entity.

BMP files are lossless and uncompressed by default [14]. They retain as much detail as possible and they are easy to edit in the same time. However, they are usually larger than other formats like PNG and JPEG.

2.2.3.2 PNG Format

PNG stands for Portable Network Graphic. It is a type of raster images particularly popular file among web designers because it can handle graphics with transparent or semi-transparent backgrounds. It is an open format that is supported by a variety of image editing software [15].

PNG images uses a lossless compression. That means the image won't lose any of its data when it's compressed. This is a big advantage over lossy options like JPEG files, where some information disappears in the compression process. However, lossless compression tends to make PNGs bigger files than JPEGs, so they require more storage space.

2.2.3.3 JPEG Format

JPEG stands for Joint Photographic Experts Group, an international organization that standardized this format during the early 1990s [16]. It's a well-known file format for digital images and it is used heavily by photographers and on web pages and it is compatible with most browsers and applications.

JPEG format uses a lossy compression where all colours that are not distinguishable by the human eye are discarded [16]. This allow images to be very small in size without a clear visual difference. This makes JPEG the preferred format for many applications. However, this small file size advantage comes at a cost of losing some details.

2.3 Image Encryption Schemes

Modern Image encryption techniques can be divided into 3 general categories: spatial, transform and mixed domain. Each category consists of a variety of techniques and sub-categories. However, almost all image encryption algorithms can be classified in one of those global types.

2.3.1 Spatial domain encryption

Spatial domain based techniques are techniques that rely on direct image pixels' manipulation by changing pixels' positions and values. These categories have known a wide interest in the recent years due to their good performance and simplicity.

Spatial based image encryption schemes are generally composed of two phases: permutation and diffusion. The permutation is the process of changing pixels' positions which eliminates the correlation between adjacent pixels. On the other hand, the diffusion phase changes the value of each pixel of the image in a pseudo-random way. The combination of these two phases renders the image pseudo-random both visually and statistically. Figure 2-9 illustrates a very common model of spatial image encryption algorithms.

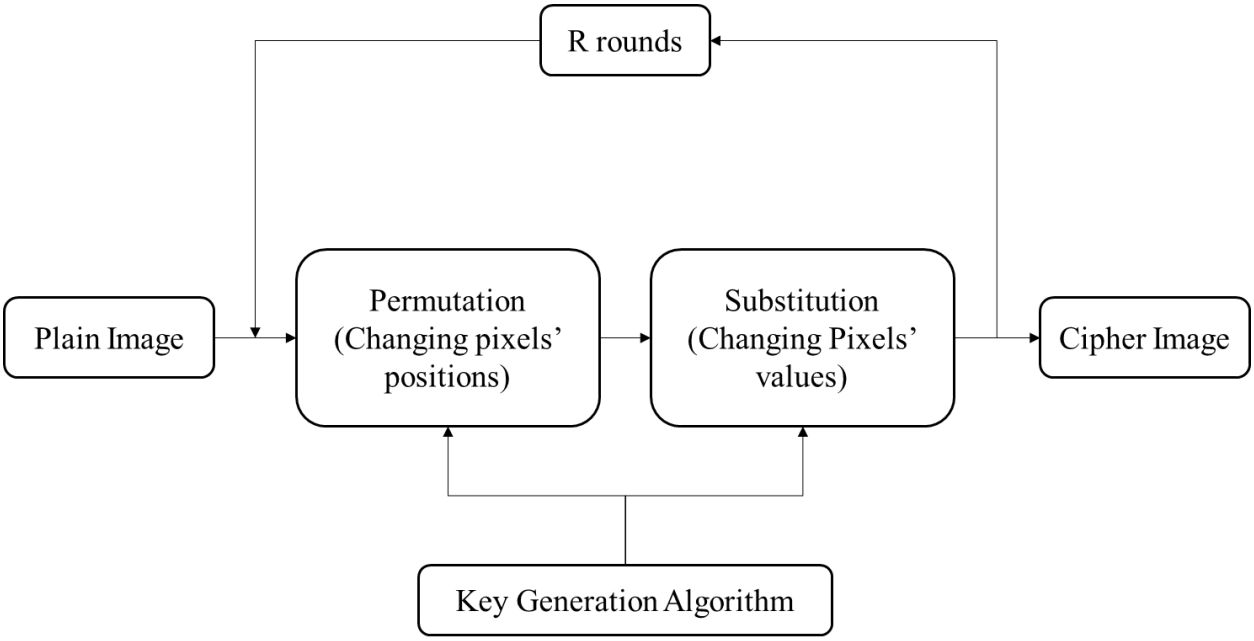


Figure 2-9: Image encryption Permutation-Substitution architecture overview

many new schemes based on spatial techniques have been proposed using chaotic maps for their performance [17]–[20], DNA (Deoxyribonucleic Acid) for their parallelism and low power consumption [20]–[22], and elliptic curves for their security and small key size [23], [24].

2.3.2 Transform domain encryption

Transform based image encryption techniques have been studied extensively in the recent years too. The general principle of such techniques is to first take the image from the spatial domain into another domain where the encryption function is applied, then transform the image back to spatial domain. Some well-known and widely used transforms in recent studies are Fractional Fourier Transform (FrFT) and Discrete Cosine Transform (DCT) [25]–[28].

In practice, the necessary modifications happen in the transform domain as illustrated in Figure 2-10. The image to be encrypted is first transformed into a given transform domain. Then, it is scrambled using the proposed encryption algorithm. Finally, it is transformed back into the spatial domain again.

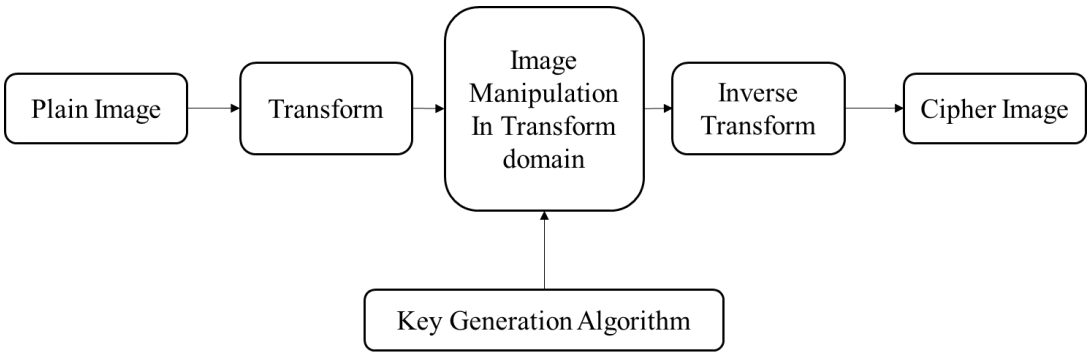


Figure 2-10: Transform domain image encryption overview

The Discrete Fourier (DFT) is one of the most used transform for image encryption especially in the optical field because of its ease of implementation [29]. Information security in the optical domain has received particular attention in recent years because optical field offers the possibility of parallel data processing resulting in high-speeds which is a desirable feature to have [30].

2.3.3 Mixed-domain encryption

Mixed domains encryption algorithms uses techniques from both spatial and transform domains. They modify the image both in spatial domain and transform domains trying to take advantage of their power to make more robust and secure schemes.

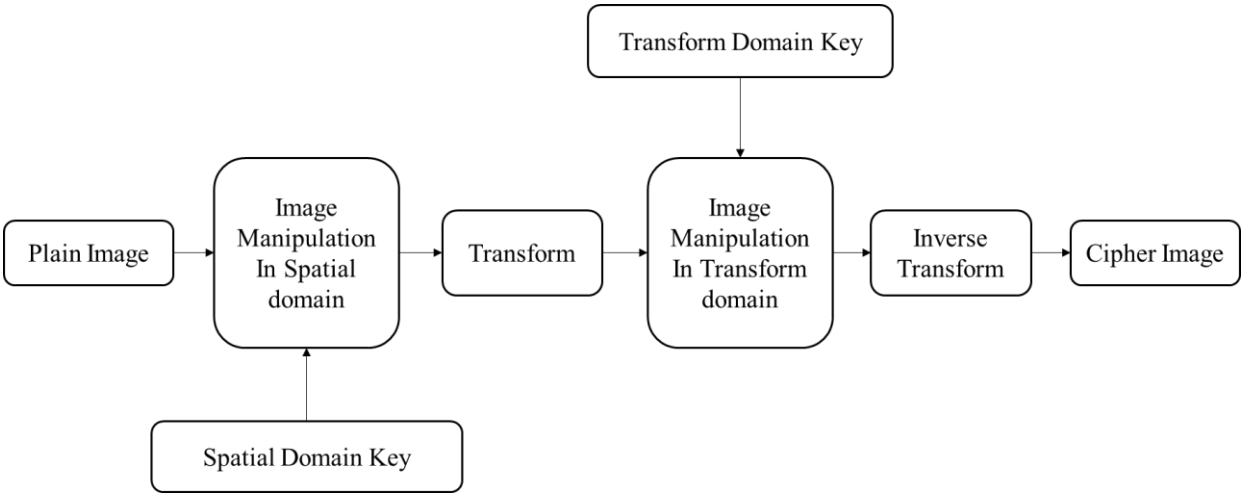


Figure 2-11: Mixed domains image encryption overview

Figure 2-11 represents one way of constructing a mixed-domain image encryption algorithm. The order of domain operations or the number of them doesn't matter. There could be multiple spatial manipulations and multiple transforms. The encryption key can be the same or it can be unique per domain. All these are implementation details that are left to the creativity and assessment of the algorithm designer.

2.4 Evaluation metrics

When a given image is encrypted, its pixels' values change compared to their original values. A good encryption algorithm should eliminate the similarity and maximize the difference in pixel values between the original and the encrypted image. Furthermore, no linearity should exist between the encrypted image pixel values. The plain and encrypted images should be completely independent and should have a low correlation in between. The cipher image patterns should be totally random and must not reveal any feature of the original image [12].

One of the important and most intuitive metrics of encryption is the visual inspection. The more hidden are the original image patterns the better the encryption algorithm is. However, visual inspection only is not reliable to judge the confusion of image patterns. Therefore, other more robust metrics should be considered to evaluate the quality of encryption quantitatively [31].

Evaluation metrics related to image encryption can be classified into two categories: metrics that measure the ability of the cipher to replace the original image with a new uncorrelated one, the other type of metrics measures the diffusion characteristics of the encryption algorithm.

2.4.1 Histogram Uniformity

An image histogram is a bar graph that represents the frequency of each gray level of the image. The horizontal axis represents the gray-level values whose range is usually from 0 to 255. While the vertical bar represents the number of occurrences of the corresponding gray level in the image [12]. The histogram of an encrypted image should satisfy two conditions:

1. It should be entirely different from the histogram of the plain image.
2. It should have a uniform distribution, which means that the probability of occurrence of any grayscale value is the same.

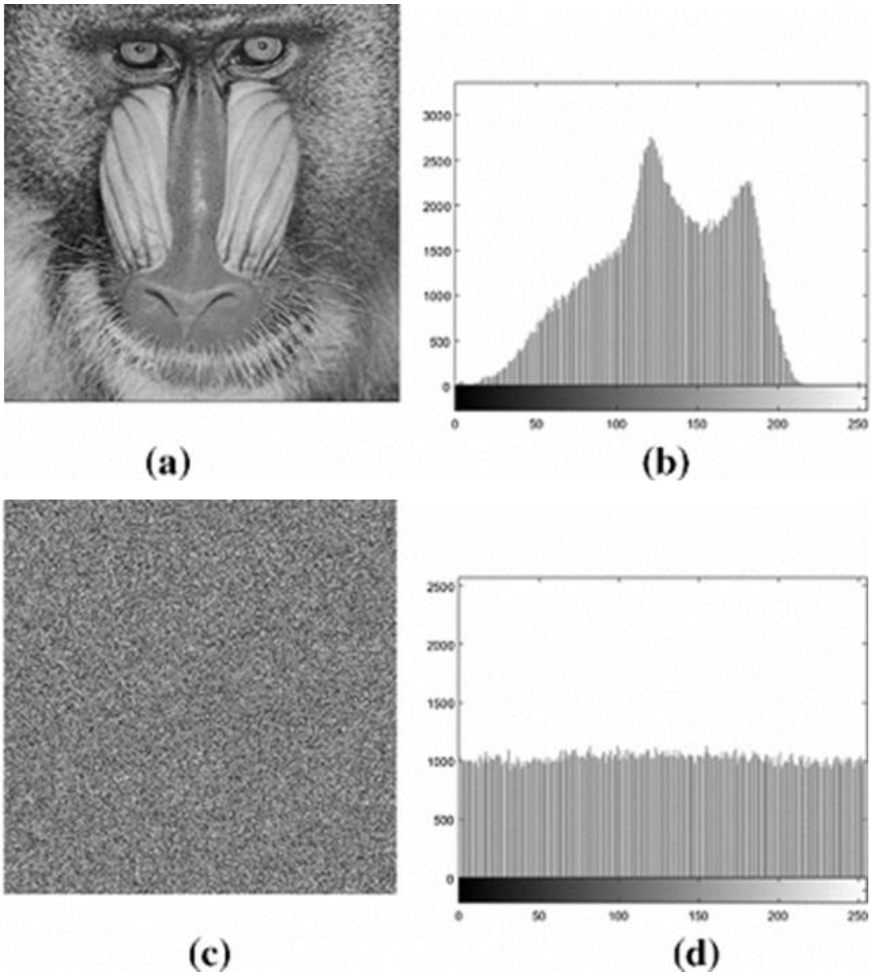


Figure 2-12: Difference between plain image and encrypted image histograms

Figure 2-12 illustrates the difference between both histograms of the plain and encrypted test image Baboon.

2.4.2 Histogram Deviation

The histogram deviation is a metric that measures the quality of encryption in terms of how it maximizes the deviation between the original and the encrypted images [32]. Histogram deviation is calculated as follows:

$$D_H = \frac{\left(\frac{d_0 + d_{255}}{2} + \sum_{i=1}^{254} d_i\right)}{M \times N} \quad \text{Eq 5}$$

Where d_i is the amplitude of the absolute difference at the gray level i. M and N are the dimensions of the image to be encrypted. The higher the value of D_H is, the better the quality of the encryption algorithm.

2.4.3 Correlation Coefficient

Correlation coefficient of pixels with the same indices between plain and encrypted image is a useful metric to assess the encryption quality of image cryptosystems [32]. It can be calculated as follows:

$$r_{xy} = \frac{cov(x,y)}{\sqrt{D(x)}\sqrt{D(y)}} \quad \text{Eq 6}$$

where x and y are the plain and cipher images. In numerical computations, the following discrete formulas can be used:

$$D(x) = \frac{1}{L}\sum_{l=1}^{L}(x_l - E(x))^2 \quad \text{Eq 7}$$

$$cov(x,y) = \frac{1}{L}\sum_{l=1}^{L}(x_l - E(x))(y_l - E(y)) \quad \text{Eq 8}$$

$E(x)$ is calculated as follows:

$$E(x) = \frac{1}{L}\sum_{l=1}^{L} x_l \qquad \textit{Eq 9}$$

where L is the number of pixels.

A low value of r_{xy}, means low similarity between the plain and the encrypted, which means a better quality of encryption.

2.4.4 Irregular Deviation

The irregular deviation measures how much the difference between the plain and encrypted image is irregular [12]. The formula for calculating this metric is as follows:

$$D_i = \frac{\sum_{i=0}^{255} H_D(i)}{M \times N} \qquad \textit{Eq 10}$$

$$H_D(i) = |H(i) - M_H| \qquad \textit{Eq 11}$$

Where H is the histogram of the absolute difference between the original and the encrypted image. M_H is the mean value of this histogram.

The lower the value of D_i, the better the encryption quality will be.

2.4.5 Deviation from Ideality

The deviation from ideality measures how close the encrypted image using a given algorithm to an assumed ideal encrypted image. An ideally encrypted image C_i must have a constant probability of existence of any gray level. Which is equivalent to a completely uniform histogram distribution, the. From this definition of the ideal encrypted image histogram, it can be formulated as

$$D = \frac{\sum_{C_i=0}^{255} |H(C_i) - H(C)|}{M \times N} \qquad \textit{Eq 12}$$

where $H(C)$ is the histogram of encrypted image. And $H(C_i)$ is calculated as according to the equation:

$$H(C_i) = \begin{cases} \frac{M \times N}{256} & 0 \leq C_i < 255 \\ 0 & elsewhere \end{cases} \quad \textit{Eq 13}$$

The lower the value of D, the closer the real encrypted image to ideality. Which means the better the encryption algorithm used to encrypt the image.

2.4.6 Avalanche Effect

Avalanche effect metric is used to test the efficiency of the diffusion mechanism of the encryption algorithm. Given an image P and a 1 pixel modified image P'. Both images P1 and P2 are encrypted to output C1 and C2 respectively. The Avalanche effect metric is the percentage of different bits between C1 and C2. If C1 and C2 are different in half of their pixels, the encryption algorithm diffusion characteristics are considered to be good. In other words, having good diffusion property means that a change of one pixel in the plain image results in a change of half image pixels or more.

2.4.7 NPCR and UACI

To test the effect of a one-pixel modification on the whole image encrypted by a given encryption algorithm, two common metrics are usually used: NPCR and UACI.

Given two cipher images C1 and C2, whose corresponding plain images have only one-pixel difference, The NPCR is defined as

$$NPCR = \frac{\sum_{i,j} D(i,j)}{M \times N} \times 100\% \quad \textit{Eq 14}$$

Where D is a binary image having the same size as C1 and C2 and calculated as follows:

$$D(i,j) = \begin{cases} 0 & C_1(i,j) = C_2(i,j) \\ 1 & C_1(i,j) \neq C_2(i,j) \end{cases} \quad \textit{Eq 15}$$

The UACI is defined as:

$$UACI = \frac{1}{M \times N}\left[\sum_{i,j}\frac{C_1(i,j) - C_2(i,j)}{255}\right] \times 100\% \qquad \textit{Eq 16}$$

The $NPCR$ measures the percentage of the number of different pixels between the two images $C1$ and $C2$ with respect to the total number of pixels, while the $UACI$ measures the average difference in intensities between the two images $C1$ and $C2$. A high value of NPCR/UACI generally results in high resistance to differential attacks. If a minor change in the plaintext image can cause a significant change in the encrypted image, then the differential attack becomes useless. And no meaningful relationship can be found between the plain image and the encrypted one [29].

2.4.8 Execution Time

The execution or processing time is the time required to encrypt and decrypt an image. The smaller the execution time, the better the encryption efficiency will be. However, sometimes achieving high level of security while maintaining a small execution time is not possible. Therefore, compromises need to be done between security and performance usually.

2.4.9 Key Space/Size

The key space of an encryption/decryption algorithm is the total number of different keys that can be used in the encryption or decryption process. While a key size is the number of bits that constitute the key. However, those two concepts are interconnected. The bigger the size of the key the bigger the key space of it. If we assume that the size of a given key is k, finding the correct key using exhaustive search will take 2^k operations.

For an image encryption scheme to be secure against brute force attacks it needs to have a key space at the order of 2^{100} at least [4].

2.4.10 Key Sensitivity

The key sensitivity is the sensitivity of the encryption algorithm to a slight change in the encryption key. It can be evaluated by encrypting an image with a given key, then making a small modification in one of the elements constituting the key and try to decrypt the encrypted

image with this new key. The decryption algorithm should produce a totally encrypted image [12].

The process above should be repeated and each time the precision of the key modification should be increased until the appearance of the original image after decryption. This limit will be the precision of this algorithm towards the encryption key.

2.5 Conclusion

In this chapter some notions about digital images are presented. The building blocks of digital images and the basic operations were discussed. Then, an overview about modern image cryptosystems was taken as well as their evaluation metrics.

Despite the effort put into developing and securing image encryption schemes. Security flaws are being constantly detected by cryptanalysts. Finding these issues in cryptosystems and highlighting them is an important step in the cycle of adapting new cryptosystems. It helps cryptosystems' designers enhancing their algorithms before they get widely adopted. This prevents many potential future security disasters from occurring.

The next chapter of this thesis will be dedicated to the presentation of a novel hybrid cryptanalysis method that we proposed against a recently published image encryption scheme.

Chapter 3

Proposed Hybrid Cryptanalysis Method

Index

Chapter 3

Proposed Hybrid Cryptanalysis Method

3.1 Introduction

Recently, a new image encryption algorithm was proposed [33]. It uses a modified version of the Arnold's chaotic cat map to permute the pixels. The resulted image after that is confused with a secret value selected randomly from the permuted image. Finally, the output image is hidden in another image selected as part of the secret key. Given that this key image can be any random image existing in the world, the authors claimed that this algorithm has an infinite key space which increases its efficiency against brute-force attacks. Therefore, it is supposed to be secure for real life applications.

The idea of a cryptosystem having an infinite key-space was interesting for us as it was not very known in the cryptographic community. We have done a thorough analysis of that algorithm trying to identify any weaknesses or security issues. Our investigation revealed that the algorithm has some serious vulnerabilities that make it completely broken and unsecure to be used with real data.

This chapter will describe our cryptanalysis work conducted against the former encryption scheme and discuss the algorithm that we proposed to completely break that algorithm encryption [34]. In the rest of this chapter we will first describe the algorithm under study. Then, a detailed cryptanalysis algorithm of the proposed scheme is presented. The experimental results showing the effectiveness of our proposed cryptanalysis are presented in the next section. Some concluding remarks are given at the end of the chapter.

3.2 Encryption Algorithm Description

The novelty of the image encryption scheme proposed by Anwar et Meghana in [33] was the use of a random image as part of the secret key. Conventional image cryptosystems transform images into noise like images. According to authors, this was a clear sign for attackers that there is some information hidden inside this image and could be a big attractor for them to try to break it. This new algorithm transforms the plain image into a visually meaningful encrypted image. Authors said that having such operation in place will not just make the encryption more secure but it may even reduce the number of attacks on the image.

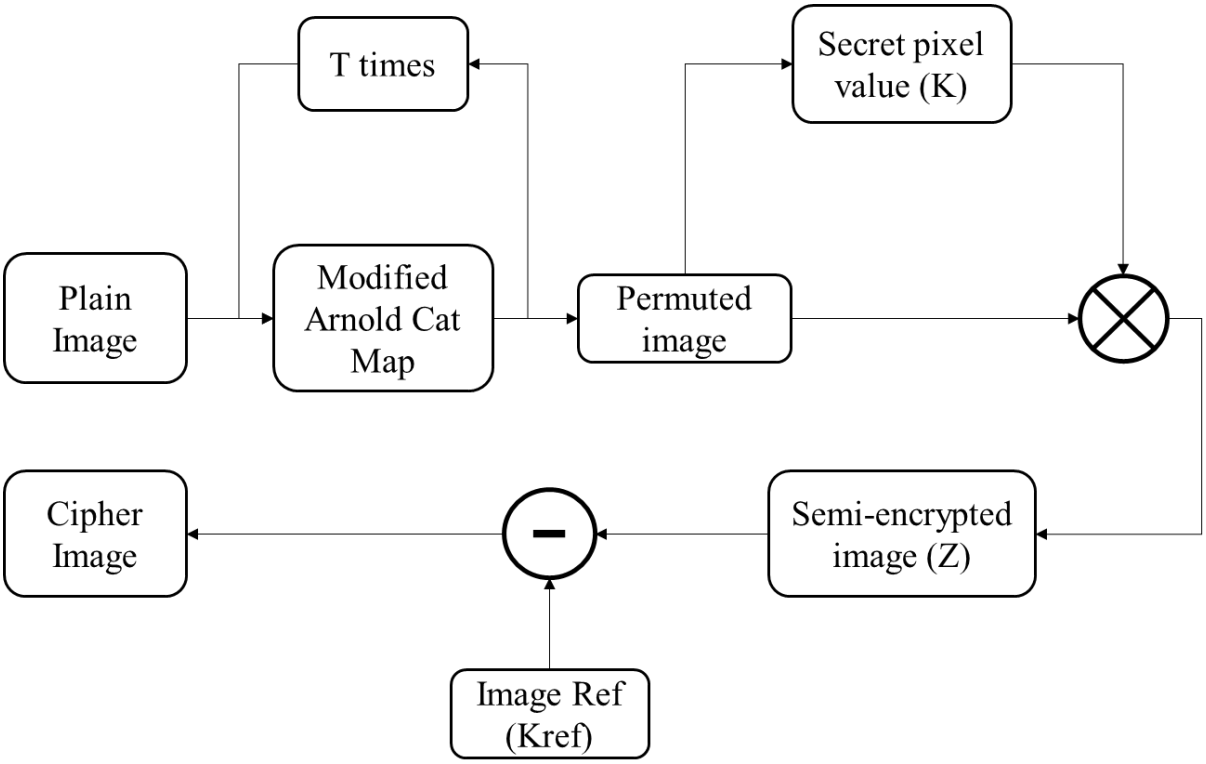


Figure 3-1: The proposed encryption algorithm diagram

Figure 3-1 represents the proposed algorithm diagram. This algorithm is based on pixel permutation using a variation of the Arnold's chaotic cat map algorithm, where each pixel is replaced by another pixel from the image. The pixel permutation is performed as shown in equation Eq 17.

$$\begin{bmatrix} x_i \\ y_i \end{bmatrix} = \begin{bmatrix} 1 & \varepsilon \\ \varphi & \varphi\varepsilon + 1 \end{bmatrix} \begin{bmatrix} x_{i-1} \\ y_{i-1} \end{bmatrix} (mod\ n) \qquad \textit{Eq 17}$$

where x, y are the i^{th} pixel position of a $n \times n$ image. Parameters ε and φ are positive constant parameters chosen such that the determinant below is equal to 1.

$$\begin{vmatrix} 1 & \varepsilon \\ \varphi & \varphi\varepsilon + 1 \end{vmatrix} = 1 \qquad \textit{Eq 18}$$

This permutation process is executed T times ($T \geq 1$). The other parameters ε and φ are chosen such that the matrix determinant is equal to 1. All those parameters T, ε and φ are parts of the secret key and they are only known to the two sides of the cryptosystem and not exposed publicly. The obtained image after permutation is then normalized to have pixel values between 0 and 255 if needed.

From that normalized image N, a random pixel whose position is part of the encryption key is chosen to use its value K a secret key. Then, a new matrix P having the same image dimensions and initialized by K is created, $P(i,j) = K$.

To obtain a semi-encrypted image Z, a bit-wise XOR operation is then performed between the normalized image N and the newly created matrix P initialized with the secret value K. This semi-encrypted image is then masked into another image K_{Ref}, called the key image. It should be chosen to be bigger or equal in size to the image that is being encrypted. This last step of hiding the encrypted image into another image acts as a steganography step that aims to elude the attackers.

$$\begin{aligned} Z(i,j) &= N(i,j) \oplus P(i,j) \\ E(i,j) &= K_{Ref}(i,j) - Z(i,j) \end{aligned} \qquad \textit{Eq 19}$$

Theoretically, the proposed algorithm has an infinite key space because any random image can be chosen as part of the secret key. Therefore, it is very efficient against brute-force attacks aiming to decode the encrypted image by an exhaustive search for the possible choices in the key space.

Figure 3-2 represents the proposed encryption algorithm written in pseudocode as appearing in the original paper [33].

Input: Image I of size $r \times c$ and the key image K_{Ref}
Output: Encrypted Image E
Intialization: Choose Period T, φ, ε such that $\begin{vmatrix} 1 & \varepsilon \\ \varphi & \varepsilon\varphi + 1 \end{vmatrix} = 1$
for k = 1 to T **do**
  for i = 1 to r **do**
    for j = 1 to c **do**
      $r' = \text{mod}((x_i + \varepsilon \times y_j), \text{rows})+1$
      $c' = \text{mod}((\varphi \times x_i + (\varepsilon\ \varphi+1) \times y_j), \text{rows})+1$
      pixel(i, j) = pixel(r′, c′)
    end for
  end for
end for
Image M is obtained after pixel permutation
N = M *255;
Randomly select any pixel position (x, y) as the Key K
Generate a matrix P of size r× c such that all values are intialized to K
Z = N⊕P
for i = 1 to r **do**
  for j = 1 to c **do**
    $E(i,j) = K_{Ref}(i,j) - Z(i,j)$
  end for
end for

Figure 3-2: Proposed Encryption Algorithm in Pseudo-code

For the decryption process, the encrypted image, the key image and the key has to be shared. The decryption process begins with using the key image and the received encrypted image to obtain an intermediary image Z. The obtained image is then XORed with secret pixel value received as part of the key. The final step is performed by inversing the permutation.

Input: Encrypted Image E of size $r \times c$, P obtained by initializing array with key K and the key image K_{Ref}
Output: Original Image I
for i = 1 to r **do**
  for j = 1 to c **do**
    $Z(i,j) = K_{Ref}(i,j) - E(i,j)$
  end for
end for
N = Z⊕P
for k = 1 to T **do**
  for i = 1 to r **do**
    for j = 1 to c **do**
      $I(x_i,y_j) = \begin{vmatrix} 1 & \varepsilon \\ \varphi & \varepsilon\varphi + 1 \end{vmatrix}^{-1} .N(x_i,y_j)$
    end for
  end for
end for

Figure 3-3: Proposed Decryption Algorithm in Pseudocode

Figure 3-3 represents the decryption algorithm written in pseudocode taken from the original paper.

Some evaluation metrics were calculated for the proposed algorithm and compared to the Arnold's Cat Map Algorithm (ACM) which is a permutation only algorithm. The metrics shows that ACM has smaller correlation coefficient while the proposed algorithm has better Structural similarity index (SSIM) and NPCR. The results are summarized in table 2.

Table 2: Proposed Encryption Algorithm metrics

	ACM			**Proposed**		
	SSIM	**CC**	**NPCR**	**SSIM**	**CC**	**NPCR**
Baboon	0.1379	0.643	97.8	0.1505	0.4852	98
Sample	0.2792	0.9936	98.8	0.7752	0.8762	99.3

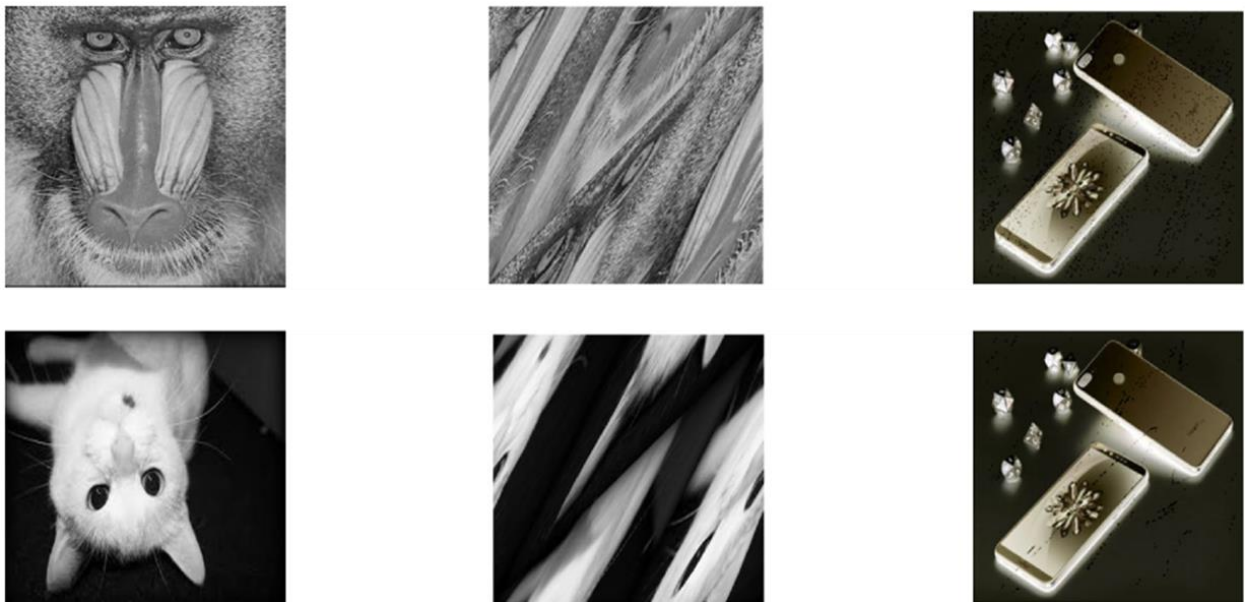


Figure 3-4: Examples of Images Encrypted Using the Proposed Algorithms

Figure 3-4 represents some examples of images encrypted using the proposed algorithm. It is clear that the resulting cipher images look similar although the plain images are different.

Because of the infinite key space and changing both pixels' values and locations during the encryption process, the authors claimed that this algorithm would be resistant to known attacks and has a high level of security.

3.3 Cryptanalysis of the proposed algorithm

An attentive investigation of the proposed algorithm reveals some security issues. These vulnerabilities will be exploited to break this cryptosystem and decode the encrypted image. In this section, we will present our proposed cryptanalysis algorithm that leads to breaking the former encryption scheme. The cryptanalysis uses a hybrid which consists of a combination of chosen-plaintext and brute-force attacks and consists of 4 main steps as follows:

1. Revealing the Key Image
2. Unmasking The Semi-Encrypted Image
3. Reversing The Permutation Phase
4. Brute-forcing the encryption key and recovering the plain image

These steps are performed in the reverse order of encryption. We start by retrieving the key image back using a chosen-plaintext attack. Then we extract out the semi-encrypted image. After that the permutation phase is reversed through a series of chosen plaintext-attacks. Finally, the secret value is deduced by a brute-force attack and the original image is retrieved successfully without the need to know anything about the secret key. The steps above are detailed in the following sections.

3.3.1 Revealing the Key Image

The strongest part of this algorithm is the use of a key image that could be any existing random image. This allows the algorithm to have an infinite key-space according to its authors. This can be true to some extent. However, in reality, the key image is easily identifiable when looking to the cipher image. Actually, they are nearly the same in most cases. This reveals some information about the used key. This leakage of information about the secret key should not happen in secure cryptosystems. In the case when the key image is an image that is available online, it will be just an image-search away.

Even in cases where the key image can’t be looked up online, the algorithm is not taking full advantage of the large key space that the key image offers. It may be able to resist brute-force attacks, but it is weak against chosen-plaintext attacks. The exact key image can be restored completely using a single chosen plain-image.

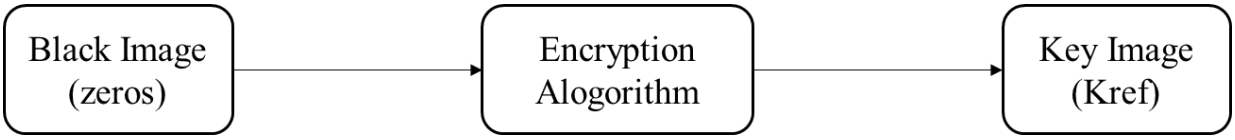


Figure 3-5: Diagram of revealing the Key Image

Figure 3-5 shows how the Key Image is revealed. According to the algorithm, if a totally black image is encrypted, the result will be the exact key image following the equation Eq20.

$$K_{Ref} = encrypt\big(zeros(m,n)\big) \qquad \textit{Eq 20}$$

In the encryption process, the permutation phase is effectless toward totally black images because all pixels' values are equal. In addition, the secret key chosen as a random pixel's value of the permuted image will be equal to 0 because all pixels' values of the images are zeros. Hence, the diffusion phase will have no effect too because 0 is a neutral element in the XOR operation. In this case, the encrypted image will be the result of subtracting a black image from the key image. Therefore, the encrypted image will be equal to the exact key image provided as a secret key to the encryption algorithm.

3.3.2 Unmasking The Semi-Encrypted Image

Since the key image K_{Ref} is already revealed from the previous step, it is possible to unmask the semi-encrypted image $Z(i,j)$ according to equation Eq 21.

$$Z(i,j) = K_{Ref}(i,j) - E(i,j) \qquad \textit{Eq 21}$$

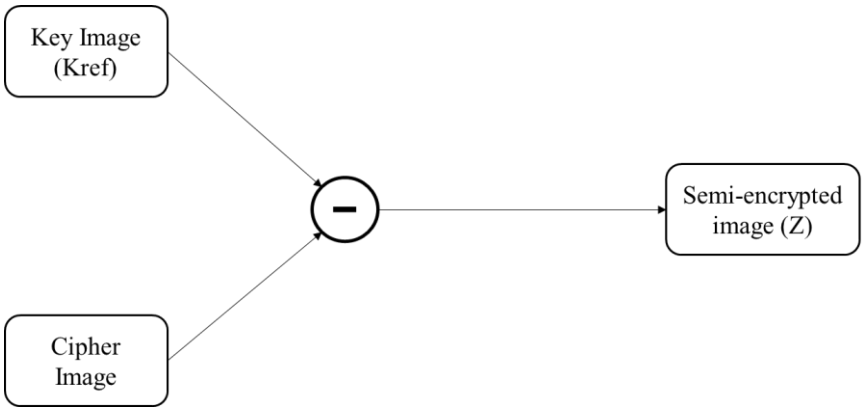


Figure 3-6: Diagram of revealing the Semi-encrypted image

Figure 3-6 represents a visual representation of the equation Eq21 above. The key image has already been revealed from the prvevious step, the cipher image is publicly known. Therefore, a pixel by pixel subtraction of the cipher image from key image will reveal the semi-encrypted image Z.

3.3.3 Reversing The Permutation Phase

The permutation phase in the proposed algorithm is based on Arnold's cat map. Without knowing its parameters and the number of times it has been executed, it is hard to reverse it using a brute force attack. However, regardless of the parameters being used and how many times the permutation operation is executed, it is possible to model that whole process as one permutation operation using a permutation vector PV following Eq22.

$$\begin{aligned} I &= reshape(I, 1, m \times n) \\ I(i) &= I\big(PV(i)\big) \\ I &= reshape(I, m, n) \end{aligned} \qquad \textit{Eq 22}$$

In this model, if the permutation vector is known, the permutation process can be easily reversed. Actually, using a series of chosen-plaintext attacks, the exact permutation vector that was used in the encryption process can be restored completely.

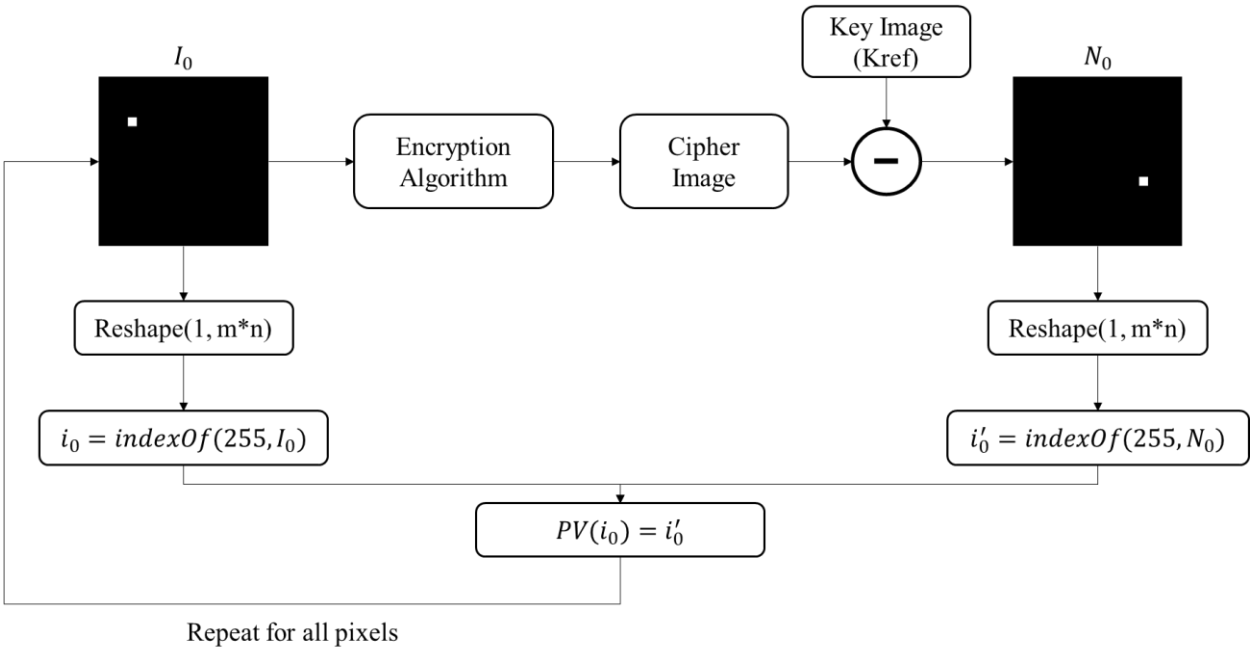


Figure 3-7: Diagram of the process that reveals the permutation vector

Figure 3-7 represents the diagram of the chosen plaintext attack algorithm used to get the permutation vector back. Given an image I_0 of size $(m \times n)$ chosen such that the pixel $I_0(0,0) = 255$ and zeros elsewhere. E_0 is noted as the encrypted image corresponding to the plain image I_0. Knowing both the key image K_{Ref} and the encrypted image E_0, the semi-encrypted image Z_0 corresponding to I_0 can be extracted using Eq21.

We know from Eq19 that $Z_0 = N_0 \oplus P_0$. In the actual case, there will be just two possible values of P_0, $P_0(i,j) \in \{0,255\}$ with 0 having a much higher probability because it is randomly selected from the permuted image P_0.

If a secret value of 0 is chosen, the diffusion phase will have no effect on the permuted image and we will get $N_0 = Z_0$. Otherwise, a value of 1 will inverse the pixels of N_0. The resulting Z_0 will be a totally white image with a single black pixel. In this later case N_0 is obtained by XORing Z_0 with 1 again.

Now that we have both the plain image I_0 and the corresponding permuted image N_0. The permutation positions i_0 (the index of 255 inside I_0) and i_0' (the index of 255 inside N_0) can be determined using a simple search. Thus, the first element of the permutation vector is found.

$$\begin{aligned} I_0 &= reshape(I_0, 1, m \times n) \\ N_0 &= reshape(N_0, 1, m \times n) \\ i_0 &= indexOf(255, I_0) \\ i_0' &= indexOf(255, N_0) \\ PV(i_0) &= i_0' \end{aligned} \qquad \textit{Eq 23}$$

The other permutation vector elements can be restored by repeating this process $(m \times n)$ times using different images I_{ixj}each time, where the pixel $I_{i \times j}(i,j) = 255$ and zeros elsewhere, $i \in \{0,\dots,m\}$ and $j \in \{0,\dots,n\}$.

Once the permutation vector is restored, the permutation phase can be reversed as shown in Eq24 below.

$$\begin{aligned} I &= reshape(I, 1, m \times n) \\ I\big(PV(i)\big) &= I(i) \\ I &= reshape(I, m, n) \end{aligned} \qquad \textit{Eq 24}$$

3.3.4 Brute Forcing the Encryption Key

At this stage, revealing the original image is just one step ahead. Knowing the semi-encrypted image Z and the permutation vector from the steps above, the permutation phase can be reversed. The result is the XOR output of the plain image and the secret key.

$$I \oplus P = reversePermutation(Z) \quad \textit{Eq 25}$$

The resulting image may not be the exact original image, However, it can be visually identifiable because all pixels are XORed with the same value. This can be sufficient in some cases where the attacker doesn't care too much about the image details.

However, because the secret key is a random pixel value from the permuted image, it has a small key-space of just 256 possible values which are easy to brute-force. By iterating over the 256 possible values of the secret key P, 256 samples will be generated. One of those samples is the exact original image that was encrypted with the algorithm. The other samples are more or less similar to the original one. Depending on the details' importance for the attacker, the number of samples can be reduced.

If the secret key value happens to be 0, the diffusion phase will not have any effect. There will be no need for this step at all because the previous step will output the exact original image. This situation is more likely to happen when encrypting images containing a lot of black areas.

3.3.5 Cryptanalysis Time Complexity

The time complexity of the proposed cryptanalysis algorithm is directly proportional to the size of the image. The more pixels the image have, the longer the algorithm will take to break its encryption.

Since the cryptanalysis scheme is based on multiple runs of the encryption algorithm itself, we can measure the time needed for cryptanalysis as a function of the image size (m, n) and the time taken by the encryption algorithm to encrypt that specific image (T_{enc}). The cryptanalysis time (T_{attack}) can be expressed approximately using the following formula:

$$T_{attack} \approx ((m * n) + 257)T_{enc} \quad \textit{Eq 26}$$

The cryptanalysis algorithm needs one encryption operation to reveal the key image, $(m * n)$ encryption operations to retrieve the permutation vector and another 256 runs to brute force the secret value used for diffusion.

3.4 Results and Discussion

To verify the effectiveness of the proposed cryptanalysis, some experiments have been executed on the grayscale image "Cameraman" of size 256×256 as shown in Figure 3-8 (a). This image is encrypted with the algorithm described in [33] using "baboon" image as the key image shown in Figure 3-8 (b). The corresponding cipher image is shown in Figure 3-8 (c).

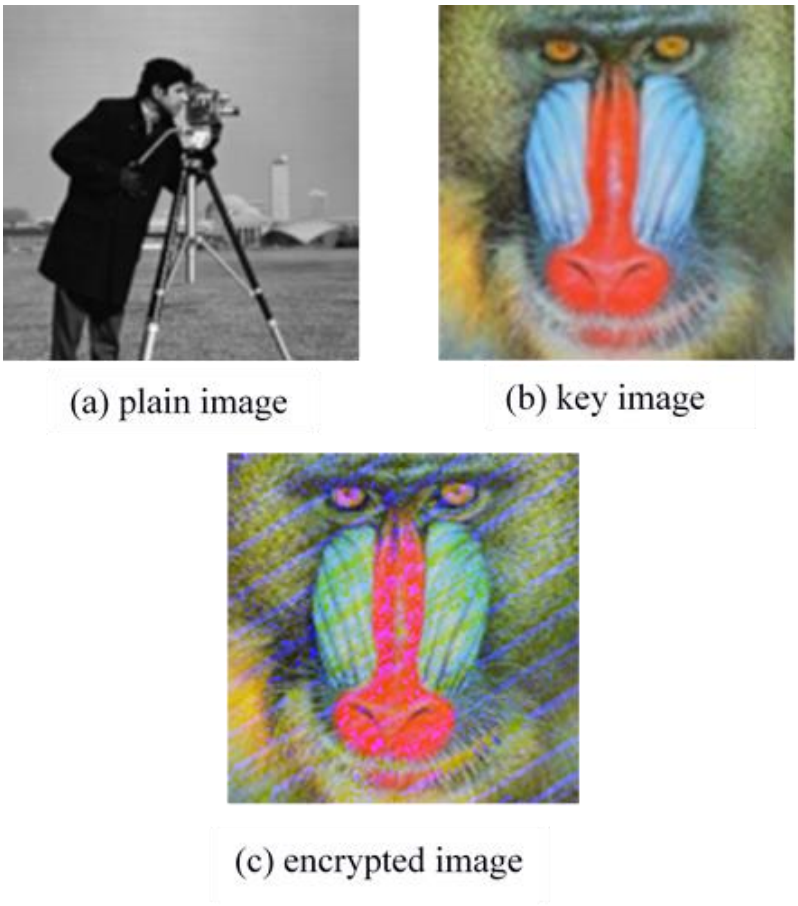


Figure 3-8: Image encryption using the proposed algorithm

In our example, we've chosen to mask the semi-encrypted image inside the blue channel of the RGB key image.

To recover the key image, a totally black image shown in Figure 3-9 (a) is encrypted using the previous encryption process. Clearly, the corresponding encrypted image shown in Figure 4-9 (b) is the Key Image used as a secret key by the encryption algorithm.

Figure 3-9: Revealing the key image using a chosen-plaintext attack

Because the used key image in this example is an image that could be found online (baboon), we tried a reverse image search using Google search engine. Because of the strong similarity between the key image and the encrypted one, Google reverse image search was able to guess the original image perfectly. The result is shown in Figure 3-10. In this case, the search result image can be downloaded and used to recover the original plain image following the rest of the steps mentioned above.

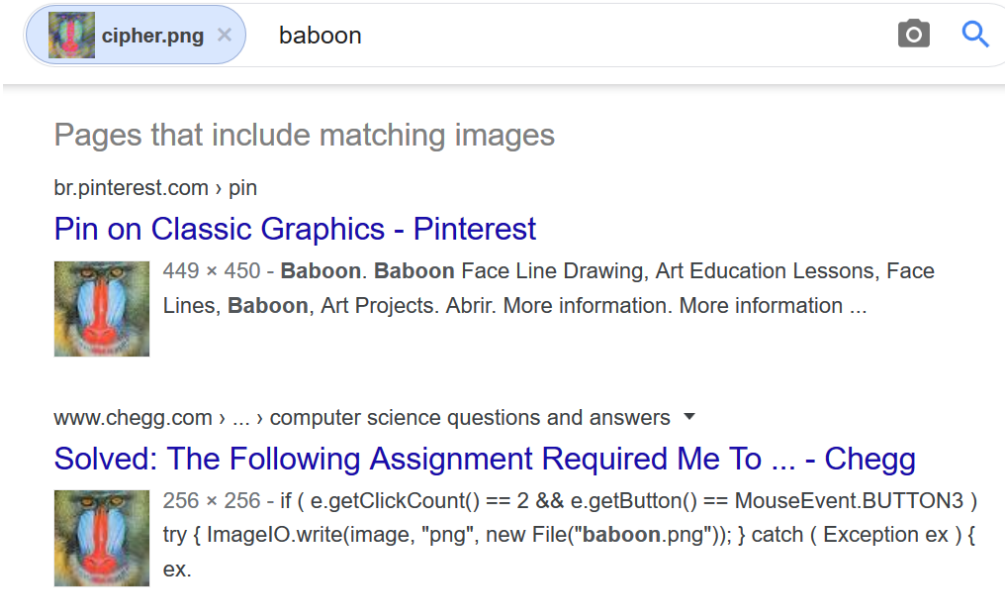
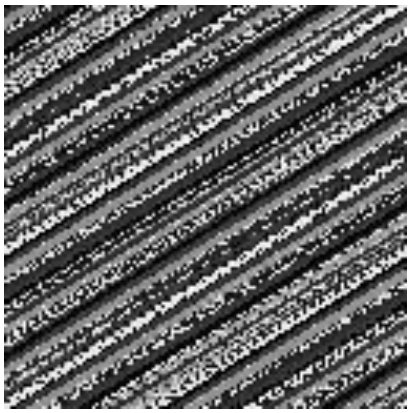


Figure 3-10: Revealing the key image using an online reverse image search using Google

Sure, this method is not going to work perfectly in all cases. It just works when the exact key image is available online and a big level of similarity exists between it and the resulting cipher image. The previous method is always preferred over this one.



(a) semi-encrypted image

(b) diffused image

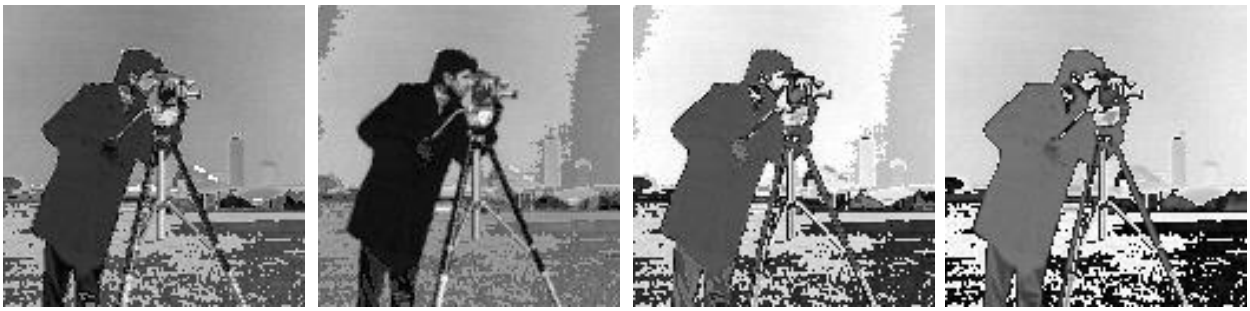
Figure 3-11: Image cryptanalysis using the proposed method

The Figure 3-11(a) shows the semi-encrypted image after its extraction from the encrypted one. After reversing the permutation phase, the image shown in Figure 3-11(b) is obtained. This image is the result of the XOR operation between the plain image and the secret value K which is a random pixel value chosen from the permuted image. Although it is not the exact same plain image that have been encrypted, it is clear enough to identify it and know for sure that it represents the "Cameraman" image.

Because "Cameramen" is a well-known test image, it was easy to guess without going much further. But, if the encrypted image was for a less known image or contains some text, it won't be easy to exploit in this form. For that reason, brute-forcing the Key will be required.

To recover the original image in a pixel perfect format, trying the 256 different cases of the key is required. For illustration purposes, just 8 samples are going to be generated as shown in Figure 3-12. The corresponding key values that will be used are $K \in \{0,32,64,96,128,160,192,224\}$.

(a) K = 0 (b) K = 32 (c) K = 64 (d) K = 96



(e) K = 128 (f) K = 160 (g) K = 192 (h) K = 224

Figure 3-12: Brute-forcing the encryption key

Figure 3-12(e) where $k = 160$ represents the most accurate generated sample compared to the original image. This means that the exact key value is not too far from 160. If more details are needed, the attaquer can easilly investigate the other values and generate all the 256 possible values.

3.5 Summary of Algorithm Weaknesses

In this chapter we performed an attack against a novel image encryption algorithm recently proposed in [33] and announced to be secure for real data encryption. Following are some of the most important weaknesses identified in the proposed scheme that made it totally breakable:

- In this encryption scheme, the cipher image has a lot of similarities with the key image. In other words, the cipher image leaks information about the secret key, which is a vulnerability that should never exist in a cryptosystem.
- Making multiple permutation only rounds doesn't offer any further improvements to the algorithm security. Any number of permutation rounds can be modelled with a single permutation vector at the end.

- Because the permutation key is chosen as a random pixel value from the permuted image, it has a reduced key space that can be brute-forced easily. Another issue with this approach is the possibility of the random pixel value to be 0. Thus, the diffusion phase will have no effect at all.
- Having just one round of encryption and a weak dependency on the plain image creates some linearity between the plain and the encrypted image, which is a sign of an unsecure algorithm.

3.6 Conclusion

After performing an attack exploiting the vulnerabilities found in former scheme, the algorithm is found to be weak against some well-known types of attacks, namely chosen-plaintext and brute-force attacks. Our proposed cryptanalysis method and experimental results proved the security claim of that scheme to be wrong. We proved that Images encrypted using this scheme can be easily decoded using our proposed method that uses a combination of chosen-text and brute-force attacks. Therefore, relying on this cryptosystem to encrypt images containing sensitive information is not secure.

In the next chapter we will present a comparative analysis that was carried between AES and a chaotic image encryption scheme recently proposed. This study was carried out in the process of building a security layer for a real web application where image encryption was needed.

Chapter 4

Comparative Analysis Between a Pixel-Wise Image Encryption Scheme and AES in a Web Application Context

Index

Chapter 4

Comparative Analysis Between a Pixel-Wise Image Encryption Scheme and AES in a Web Application Context

4.1 Introduction

The study presented in this chapter was carried out in the process of building a security layer for a real web application where a huge amount of user generated images need to be stored on the application servers and delivered to the right users on their request.

To ensure that images are stored securely on servers, an encryption system was required to be integrated into the application. The first encryption scheme that was proposed to be used as the application encryption layer was AES. The reason was that it is general purpose and the actual standard encryption algorithm for data. However, pixel-wise category of ciphers especially chaos based image encryption algorithms were heavily studied recently as an alternative way to traditional algorithms in the field of image encryption. This rise of their popularity made this kind of algorithms worth investigating as an alternative solution.

After some research among the recently proposed schemes, we found a chaotic encryption scheme that uses a substitution-permutation network and based on a simple Pseudo Random Number Generator called Bülban chaotic map [35]. The interesting part in that scheme was the raise of the processing unit from individual pixels to the images' rows and columns which led to less computations and more speed while maintaining the same level of security as other schemes. Due to the potential this scheme has, it was decided by the development team to be the algorithm that is going to be tested alongside AES.

This chapter will be divided into four sections. The first and the second sections will be brief descriptions of the chaotic algorithm and the relevant part of the web application respectively. The third section will present the tests that we designed based on the real needs of the application to test the proposed ciphers.

In the fourth section we will present and discuss the experimental results of comparing AES_GCM_256 cipher and the above mentioned chaotic cipher using our test suite.

4.2 Description of the Chaotic Encryption Scheme

The pixel-wise encryption model is a well-known and commonly studied type of image encryption schemes. It consists of taking a matrix of pixels as input and feed it into an encryption algorithm which randomizes the pixels according to a secret key. Then, it outputs a matrix of random-looking pixels as the encrypted image, usually of the same dimensions as the input image as illustrated in **Fig.4-1**.

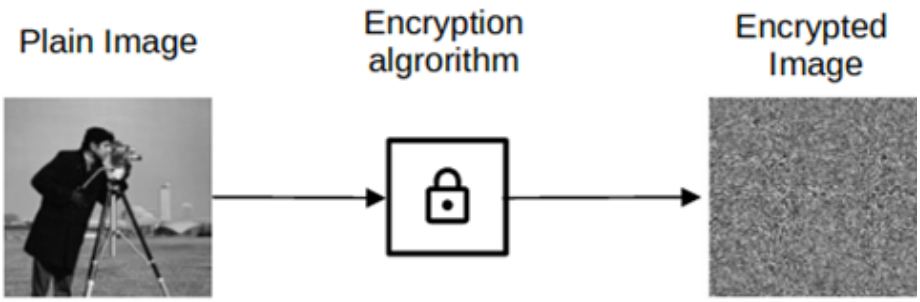


Figure 4-1: . The pixel-wise image encryption model

Algorithms following this model are generally described using a one channel grayscale image. However, they can be generalized to color images containing multiple channels by encrypting each channel separately and combine them later into one color encrypted image. Later, the decryption process is done by separating the encrypted color image channels and applying the decryption algorithm on each of them, then combine them into one image to regenerate the plain image.

Pixel-wise encryption schemes generally differ in the pixels' manipulation technique used to randomize the plain image's pixels. Recently, cryptosystems' designers and security researchers developed many techniques that differ in speed, security and implementation complexity.

However, many of them accomplish the same purpose in general, which is the randomization of the input image pixels.

These pixels' manipulation techniques can be divided into 3 general categories based on the domain where the pixel randomization happen: spatial, transform and mixed domains as mentioned in Chapter 2. However, there are other classifications based on different parameters.

Talhaoui et al proposed in [35] a new efficient and high-speed image encryption scheme based on the Bülban chaotic map originally proposed in [36] and defined by the following equation:

$$x_{n+1} = x_n * \sqrt{\frac{a}{x_n - b}} \qquad \textit{Eq 27}$$

Where, x_0, a, b are secret parameters chosen such as $x_0 > b > 0$ and $a > 0$.

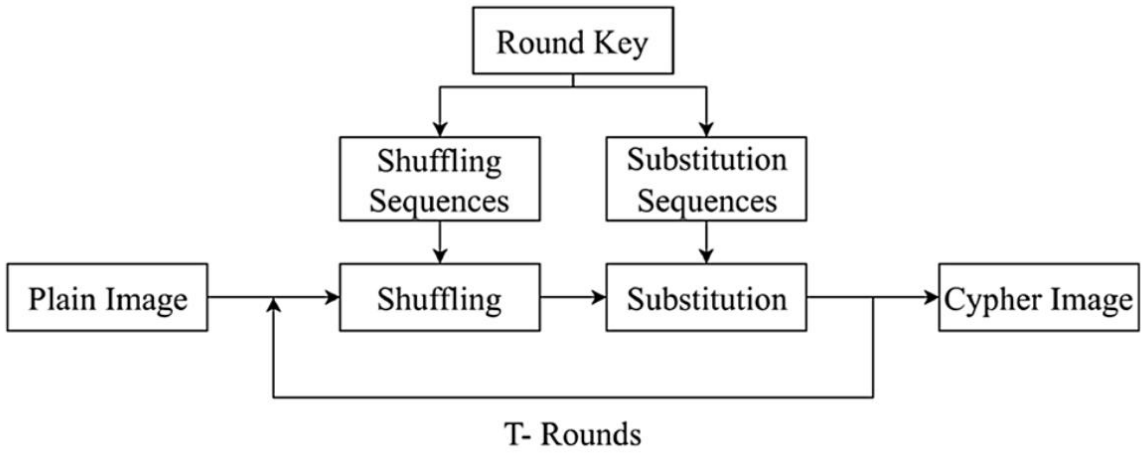


Figure 4-2: Encryption Diagram

Figure 4-2 illustrates the encryption diagram of the actual scheme. The plain image is fed into a Substitution-Permutation network where a circular shift of rows and columns is applied to break the correlation of adjacent pixels and the XOR operation with the Modulo function is used to mask the pixels' values and prevent the leak of information. The process is repeated T times and each round uses different shuffling and substitution sequences generated using the chaotic map. The output of the SP network on the last round represents the cipher image.

Some design decisions were wisely made by the authors to increase the speed of that algorithm. They intentionally choose a simple chaotic map and a simple structure. Also, the processing

unit was raised from the pixel level to the row/column level. This enhances the algorithm efficiency by requiring the generation of a small set of random numbers only.

The security tests and simulation analysis have been carried out by the authors. They stated that the scheme is extremely secure and very fast for real-time image processing.

The other cipher AES which will be compared with this scheme is described in Chapter 1.

4.3 Brief Description of the Web Application

The web application in hands is a professional network for artists, where people creates their portfolios and upload their original and creative art works and pictures and control its visibility. The application has a big reliance on users' uploaded original images. It has both web and mobile interfaces where users interact and consume data which is mostly composed of images. The two main parts of the application that are relevant to image encryption are the image uploading and serving systems.

4.3.1 Image Uploading System

The images that needs to be encrypted are the user generated ones. The image is first uploaded by the users through the web interface or the mobile application. When it reaches the server, it needs to be encrypted with a predefined key. Then the encrypted version is stored in the server storage space and a copy of its location is then stored in the database with a reference to the user that uploaded it.

4.3.2 Image Serving System

The stored images are going to be shown later to users based on some predefined permissions. When the user requested the image using an HTTP request, the server first check the user permissions, if the user doesn't have the permission to view the image, the server immediately respond with a 403 (forbidden) status code. Otherwise, it reads the image from the storage space and decrypt it using the same encryption key then sends it back. The user then views the plain image on the web or mobile application.

4.4 Application encryption requirements

In order to select the right cipher to use for image encryption within the application, we first defined the application requirements and needs. Those requirements will serve as guidelines for designing the tests later. Below are the application requirements described briefly.

4.4.1 High Security

For an application that stores users' images to be trustworthy, it needs to treat security very seriously. The uploaded images should be kept safe from unauthorized access from other users or even from hackers that may get access to the server storage space. The used encryption algorithm needs to be secure against both passive and active attacks. That means, a hacker with access to storage space should get no information from the encrypted images stored there. Moreover, if encrypted images are changed, the compromised images should be detected so they can be deleted and replaced from the backup storage. The application should never try to decrypt and send compromised images to the users. We cannot be sure if the modification is random or maliciously done, so the safest way is to discard the infected images and replace them from the backup.

4.4.2 High Performance

One of the most critical aspects for a high traffic online application is speed. In the current application, for security reasons, images are stored encrypted and no plain versions of them are kept. That means, images should be decrypted on every request to the server. A slow cryptosystem will slow down the server by making it stuck in encryption or decryption process for a long period of time. Therefore, the number of requests per second that the server is capable of handling will decrease which will directly affect the user experience when many users are connected simultaneously. This will also increase the financial dispenses needed for paying the servers very rapidly with the growth of the user base. In the current use case, cryptosystem performance especially decryption speed is very critical.

4.4.3 Side Effect Freedom

Another important requirement of the needed encryption scheme is that it needs to be side-effect free. It needs to keep all the original image information such as metadata and image's dimensions for copyright reasons. Therefore, no data loss is tolerated after encryption and decryption. Moreover, the encrypted images need to be roughly the same size as the original ones. Original images are compressed and highly optimized before encryption to save storage space. Therefore, any increase in size after encryption should not surpass 10% of the original size.

4.5 Evaluation Tests

To select the most suited algorithm for the application, we designed 7 tests evaluating different aspects of the competing candidates. Those tests were built with the real world application usage scenario in mind. They are high level end-to-end tests that don't test the internals of the ciphers themselves. In fact, they test the outcomes of the cryptosystems and their behavior from a software engineering perspective. The tests are listed and described below.

4.5.1 Eavesdropping resistance test

The eavesdropping resistance test will evaluate the ability of the cipher to hide data. This test follows these steps:

1. encrypting a test image using the encryption algorithm being evaluated.
2. Members of the team are asked to act as potential attackers and analyze the encrypted image trying to get as many information about the original image as possible.
3. All information that can be extracted are listed in a table and compared with other ciphers.

The more the quantity of information that can be extracted about the plain image from the encrypted one, the less resistant to eavesdropping the corresponding cipher is.

4.5.2 Randomization Test

A good encryption algorithm should output different results when encrypting the same data again. This is why block ciphers' modes of operation were invented and developed. Outputting the same result when encrypting similar data leaks information about the plain data that should be hidden and opens the door for differential attacks. This test evaluates the algorithm's ability to randomize the encryption of similar images under the use of the same secret key. The test consists of the following steps:

1. Take a test image and encrypt it using the encryption algorithm.
2. Repeat the encryption process again for the same image using the same secret key while changing the initialization vector or any initialization information that the algorithm support.
3. Calculate the SHA256 hash of the two encrypted images and compare them.

For this test to be considered successful, the two hash values should be different. This means the algorithm outputs different results for the same data each time it is executed. This is a desired behavior.

4.5.3 Integrity test

Data integrity is one of the goals of cryptography. Preserving data integrity means the impossibility of tampering with the encrypted data without detecting in the decryption phase that the data was manipulated by some third party. In algorithms that guarantee the integrity of data, if one bit of the encrypted data is modified, the decryption algorithm will detect it. In our case this guarantees that whatever image the user uploaded is what he will get when he requested it back later. If any sort of modifications happens to the encrypted image it should be detected before decryption. This means that the application can delete the infected image and get a valid copy of it from the backup system. Also, this will be a clear indication that some kind of hacking or incident happened and investigations must be conducted by the development team.

This test will consist of the following steps:

1. Take a test image and encrypt it.
2. Change one bit of the encrypted image.
3. Decrypt the modified encrypted image using the same key.

If the decryption algorithm detects that the encrypted data was modified, it passes the test, otherwise it will be considered a failure.

4.5.4 Symmetry Test

The symmetry test goal is to test the ability of the cipher to fully recover the data after applying encryption and decryption operations in sequence. It tests if the ciphers in question verify Eq28 which is the definition of a symmetric cipher.

$$D(E(x,k),k) = x \qquad \textit{Eq 28}$$

The test will be conducted as follows:

1. Take a picture with a digital camera.
2. Calculate the SHA256 hash of this image.

3. Encrypt the image.
4. Decrypt the encrypted image using the same key used for encryption.
5. Calculate the SHA256 hash of the decrypted image and compare it to the hash found before encryption.

The two hashes must be equal for this test to pass. Finding two different values of the SHA256 hash means that the cipher is not symmetric. Therefore, the algorithm is not 100% reversible and has low fidelity reproducing the original data.

4.5.5 Storage Space test

Storage space is a limited resource. In web applications, user-generated content and uploaded media can get too big and its storage can become too expensive if not managed properly. Having small files stored is always recommended. That's why in this application big images are resized to a resolution of 2-4 MP and optimized for storage immediately when they are uploaded, before the encryption step. Therefore, it is recommended that the used cipher outputs encrypted files of the same size as the original ones with a toleration of plus 10% at maximum.

This test evaluates the change in size between the original and encrypted images and between the original the decrypted ones. In the current use case, images are being adequately compressed and highly optimized. Therefore, it is required that the cipher doesn't break these optimizations by either increasing the image size more than 10% of the original size or degrading the image visually. The test will be conducted as follows:

1. Take a picture with a digital camera.
2. Record the file size of this image.
3. Encrypt the image.
4. Record the file size of the encrypted image
5. Decrypt the image.
6. Record the file size of the decrypted image.
7. Compare the file sizes of the original, the encrypted and the decrypted images.

For this test to pass, the encrypted image must satisfy the formula below, where $encSize$ is the size of encrypted image and $orgSize$ is the size of the original one.

$$encSize \leq 110\% \, orgSize \qquad \textit{Eq 29}$$

On the other hand, the decrypted image must be the same size as the original image.

4.5.6 Raw speed test

In our case, the cipher raw speed of encryption and decryption is directly related to the performance of the application. Therefore, it is very important for the cipher to have a good speed to avoid degrading the responsiveness of the server and sacrifice the whole user experience. This test evaluates the raw speed of the cipher's encryption and decryption processes when executed in isolation. And it goes as follows:

1. Select three test images:
    a. 256*256 grayscale image
    b. 256*256 color image
    c. 2 – 4 Mega Pixels image
2. Encrypt each image 10 times and calculate the average time.
3. Decrypt all the previously encrypted images 10 times and calculate the average time.
4. Record the average times in a table for later comparisons with other ciphers.

In this test, we tested with an image of 2 – 4 MP on purpose since it is the common resolution that is going to be encrypted after the application preprocessing of uploaded images. This size will be a good simulation of the real images that will be encrypted. It will be a more realistic test than using the common small test images only.

4.5.7 Server Load test

An important metric that defines the responsiveness of web applications is the number of requests per second that the server is capable of handling. This tests a real scenario when many users connect to the server simultaneously. This test will be performed as follows:

1. Create a simple Node.js server that reads a previously encrypted image from the storage space, decrypts it and send it in response.
2. Run this server on localhost.
3. Using load test tools such as "Autocannon" or "wrk", start a load test against the server. And repeat the test 3 times.
4. Calculate the average value of requests per second and record it in a table for comparison.

The higher the number of requests per second the more money the corresponding cipher will save the company on scale. Therefore, high values are desired.

4.6 Results and Discussions

In this section, the tests described above will be implemented over both AES and the chaotic cipher to identify which cipher is better suited to be used in the former web application.

The AES version that will be used is the AES_GCM_256 which mean AES block cipher with a key length of 256 and using Galois Counter Mode as mode of operation. This version of the algorithm comes embedded in Node.js under the "crypto" built-in library.

The chaotic cipher is going to be coded from scratch using Node.js and the "scijs" suite especially the "ndarray" package to simplify working with multi-dimensional array, "imread" and "imwrite" packages to decode image files and encode them back.

The test images that are going to be used throughout the tests are 3 images:

1. The first image is a grayscale 256*256 pixels PNG image without metadata called Cameraman.
2. The second is a color 256*256 pixels JPEG image without metadata called Baboon.
3. The third image is a color 2048*1536 pixels JPEG image captured with a phone camera and resized while keeping all the original metadata. This image will be referred to as "Minaret".

Figure 4-3: Images used in the experimental tests

Fig.4-2 illustrates the former mentioned test images. Left is Minaret, top right is Cameraman and bottom right is Baboon.

4.6.1 Eavesdropping resistance test

In this test, Baboon image is encrypted using both algorithms, AES and The Chaotic Algorithm. The result of AES is a binary file, while the result of the chaotic algorithm is an image composed of random looking pixels of the same dimensions as the original.

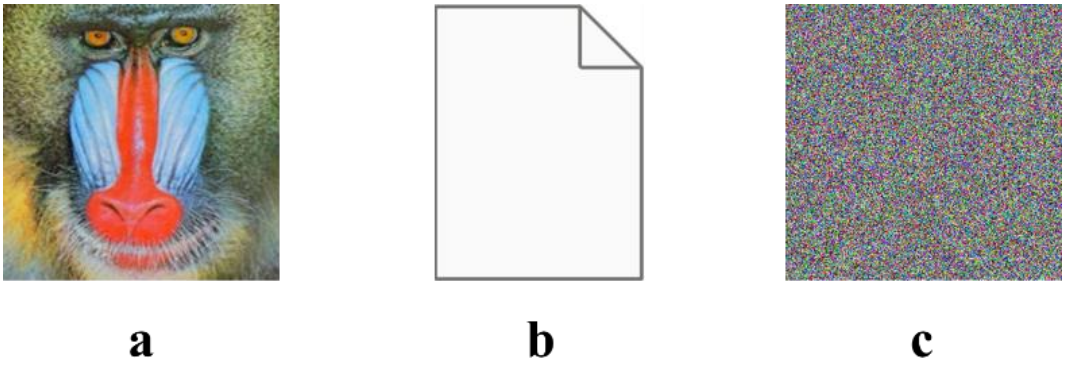


Figure 4-4: Baboon test image encryption using AES and the Chaotic cipher

Fig.4-4 represents the encryption of Baboon test image using AES cipher and the chaotic cipher. Fig.4-4 (a) represents the original image, while (b) and (c) represent the AES and the chaotic encrypted images respectively.

Using tools like the operating system file explorer and an image editing programs (like GIMP), Here are the information the team was able to extract from each encrypted file about the original image listed in **Table 3**.

Table 3: Eavesdropping resistance test results

Cipher	Leaked information
AES	Original Data size
Chaotic Cipher	Original Data type Original Image resolution Original Image dimensions Original Number of channels

We notice that AES is hiding all information about the original image other than the data size. The chaotic cipher on the other hand is revealing the original data type and exposing that the original data was an image, the image resolution (the number of pixels), the image dimensions and whether the original image was grayscale or colored by exposing the number of channels.

Clearly the chaotic cipher is exposing a lot of information about the original image. Whether an attacker can reveal the exact original content of the plain image or not, knowing only the above information about the plain data defeats the purpose of confidentiality. Cryptography insists on the fact that an attacker should know nothing about the plain data. Depending on the application, those revealed pieces of information can be very dangerous to expose publicly. They can act as a basis on which attackers can build more advanced and targeted attacks.

AES cipher is exposing less information about the original image. Therefore, it will be the winner of this test.

4.6.2 Randomization Test

In this test, the same Cameraman test image is encrypted twice using AES and twice using the chaotic cipher. On each experiment the encrypted image SHA256 hash is calculated. The test results are summarized in **Table 4**.

Table 4: Randomization test results

Experiment	SHA256 hash
AES Enc 1	K7P3AFb69NMuItxpBk+tCSrz3HlBZe79RR1ZtNE+Bgk=
AES Enc 2	b7DSTMtBz4O7naxajHn1jXjNQ0R4u6C7GGZwH3tQUyQ=
Chaotic Enc 1	ZTH/oRaYiw/dHdypwRQNNyGbmDYRiZyzLWuU576Qxb0=
Chaotic Enc 2	BOGERR4xGWEqQsoJSJg2c/SSab4tQVR6S/aVH+/uPsc=

The goal of this test is to assess the ability of the cipher to randomize the results and hide patterns created by similar images when they are encrypted.

In Table 4 we see that both AES and the chaotic cipher output different encrypted images each time they run although giving the same input image and using the same secret key.

AES_GCM_256 cipher is capable of randomizing the results using the initialization vector it accepts when initializing the encryption algorithm. This vector is a short sequence of random bits used to randomized the output when the same input is given. It got mixed with the key and passed unencrypted with the cipher data. Later, the decryption algorithm will be able to reuse the same IV used in encryption to correctly decrypt the data.

The problem of lack of randomization in pixel-wise cipher is very common and it is found in many other pixel-wise image ciphers [20], [23], [24], [37]–[45]. However, the current chaotic cipher well solved this issue. It did that by the introduction of two extra random rows on the top and bottom of the image at the first round. Those rows are included in the confusion step and due to the recursive confusion of pixels, randomness affects all the encrypted image pixels.

Both ciphers pass the randomization test.

4.6.3 Integrity test

To test the level of encrypted data integrity provided by each cipher. The Cameraman test image is encrypted twice. One time by AES and the other by the chaotic cipher. Then, both encrypted images are slightly modified. The AES encrypted image file is modified directly using a binary file editor. Since the chaotic cipher encrypted image is stored as an image file format, modifying it directly gave inconsistent behavior in decoding the file later, some bits have special significance in the file format used so it breaks the decoding operation and the encryption algorithm will not execute. Therefore, the modification will be done after decoding and before decryption to a random pixel with a random value.

Finally, both modified encrypted images are tried to be decrypted using the same key. The decryption results are summarized in Table 5.

Table 5: Integrity test results

Cipher	Modification detected	Detection method
AES	Yes	Authentication Failed
Chaotic	No	/

AES stopped before performing decryption showing a message that data authentication failed. AES_GCM_256 stores an authentication tag alongside the encrypted data, on decryption it calculates the tag of the data again and compare it with the tag of the original data. If a single bit of data is changed, the new tag will be completely different and the cipher will throw an error preventing the decryption of a manipulated data.

The chaotic cipher on the other hand failed to detect the pixel change and decrypt the image normally. The reason for that is the absence of any kind of checksums or authentication tags calculated on the original data to protect it from undetected manipulations. This lack of integrity check was found to be common in many pixel-wise ciphers [39]–[42].

AES is clearly the winner in this test.

4.6.4 Symmetry Test

To test the symmetry of the two ciphers, both 3 test images will be needed. Each of them will be encrypted by both algorithms and decrypted using the same key used for encryption. The original image and the decrypted one will be compared together using their SHA256 hashes. The experimental results are summarized in Table 6.

Table 6: Symmetry test results

	AES Cipher	
	Original Image Hash	**Encrypted Image Hash**
Cameraman	9wuhrJv36qvFde2NLS8Kk6O1Fd9RFFTWHSIV78qqUKA=	9wuhrJv36qvFde2NLS8Kk6O1Fd9RFFTWHSIV78qqUKA=
Baboon	cVgVr5Dzffso+wKqemzc5CHOx0FgcNE31znOXNyWkYc=	cVgVr5Dzffso+wKqemzc5CHOx0FgcNE31znOXNyWkYc=
Minaret	SwgK5CeHcR7vBRCzdtt0qKL2Ki1A5vtva7YVLnMNSRZ=	SwgK5CeHcR7vBRCzdtt0qKL2Ki1A5vtva7YVLnMNSRZ=
	Chaotic Cipher	
	Original Image Hash	**Decrypted Hash**
Cameraman	9wuhrJv36qvFde2NLS8Kk6O1Fd9RFFTWHSIV78qqUKA=	9wuhrJv36qvFde2NLS8Kk6O1Fd9RFFTWHSIV78qqUKA=
Baboon	cVgVr5Dzffso+wKqemzc5CHOx0FgcNE31znOXNyWkYc=	RfPehJSlZ/d/8eCoAaJBVIeroZr8n0HGCQSfpC7kqHw=
Minaret	SwgK5CeHcR7vBRCzdtt0qKL2Ki1A5vtva7YVLnMNSRZ=	P3KWnVtiDwcdAEfiH79ElJn3ETvaI0OzD2BGF9q00ZV=

It is noted that AES restored the exact same image file after encryption and decryption. The chaotic cipher on the other hand didn't return the exact original file though the exact pixel values and positions where restored. This issue in the chaotic cipher is caused by a loss of data or an encoding/decoding parameters modification.

Since the chaotic algorithm is a pixel-wise cipher, the only image data it considers is pixels' values. Image metadata are completely ignored. Therefore, the encrypted image will lack the metadata that existed on the original one and so will be the decrypted image, and this issue is common in many image ciphers proposed recently [17]–[26], [46], [37], [47], [38], [48], [49], [39]–[45], [50]–[60]. In decryption, if the image is encoded into another image format than the original image or used different output parameter like JPEG quality or PNG compression ratio, the decrypted image will differ too from the original one.

To have a decrypted image the same as the original one in a pixel-wise cipher, these 3 conditions needs to be satisfied:

- Original image should not have metadata
- Encrypted image should be saved in a lossless image format.
- Decrypted image should be encoded in the same original format and using the same parameters

Satisfying these conditions is hard and impractical in real world applications. Therefore, the chaotic cipher has failed this test and the winner is AES.

4.6.5 Storage Space test

To test the effects of using both ciphers on storage space. The 3 test images will be encrypted and decrypted again using both ciphers. On each step the file size is recorded. The results are summarized in Table 7.

For AES, we see that the file sizes are consistent, encrypted images are always bigger than their original one by a fixed value which is the size of initialization vector plus the size of the authentication tag attached to the encrypted data. In this case the increase in size is equal to 0.03 KB and it is much less than the tolerated 10% of the original size.

For the chaotic cipher on the other hand, an inconsistency in file sizes is noted. All images are encrypted and saved in PNG format to compress image while keeping the data. The decryption

is done back to the original image format: Cameraman is restored back to PNG while Baboon and Minaret are encoded back to JPG.

Table 7: Storage space test results

AES Cipher			
	Original Size (KB)	**Encrypted Size (KB)**	**Decrypted Size (KB)**
Cameraman	37.87	37.90	37.87
Baboon	37.91	37.94	37.91
Minaret	691	691.03	691
Chaotic Cipher			
	Original Size (KB)	**Encrypted Size (KB)**	**Decrypted Size (KB)**
Cameraman	37.87	218.56	125
Baboon	37.91	219.52	18.25
Minaret	691	10511.73	393.33

We see a clear increase in the size of encrypted image, because this pixel randomization breaks the correlation between the original image pixels and eliminates similar patterns. Therefore, it reduces the effect of lossless compression on the encrypted image. As a result, the encrypted image size will most likely be bigger than the plain image one.

Table 8: Summary of encoding and metadata effects on decrypted image size

	Has Metadata	**No Metadata**
Same encoding Parameters	Different size	Same size
Different encoding Parameters	Different size	Different size

The size of decrypted images compared to the original size depends mostly on the encoder and existence of metadata. The effect of the encoder parameters and metadata existence on the pixel-wise ciphers' decrypted file size are listed in Table 8.

The only case where the chaotic cipher outputs a decrypted image having the same size as the original one is when the original image has no metadata and the same parameters used to encode the original image are used to encode the decrypted image too.

AES clearly wins this test.

4.6.6 Raw speed test

Raw speed test will evaluate the speed of both ciphers executed in isolation on a set of test images. The test consists of executing for each cipher 10 rounds of encryption and decryption over the 3 test images. Then the average time for each experiment is calculated and recorded. This test and the next one were performed on a HP OMEN 15 laptop equipped with a Ryzen 7 5800H CPU and 16 GB or RAM. The test results are summarized in Table 9.

Table 9: Encryption and decryption speed results

AES Cipher		
	Encryption speed (ms)	**Decryption speed (ms)**
Cameraman	1.91	1.05
Baboon	0.76	0.51
Minaret	2.78	3.08
Chaotic Cipher		
	Encryption speed (ms)	**Decryption speed (ms)**
Cameraman	145.02	98.45
Baboon	672.77	458.82
Minaret	3754	4536

We see in the table above that AES outperform the pixel-wise cipher in terms of raw speed by orders of magnitude. AES is ~94x to 1400x faster. This ratio is directly related to the number of the image pixels, the more pixels the image has the slower will be the pixel-wise cipher even if the file size is the same like in the case of Cameraman and Baboon. They have approximately the same file size ~38 KB and the same dimensions of 256*256. However, baboon is an RGB image so it has 3 channels instead of 1. Therefore, the time needed to encrypt of decrypt Baboon is roughly ~3 times the time needed for Cameraman.

The first reason of AES outperformance of the chaotic cipher is the nature of the pixel-wise algorithms in general. Dealing with pixel data is less efficient and less predictable in terms of speed than dealing with binary data directly. In addition to the complex mathematical operations used inside operations like square roots and powers of big numbers. While in AES, all the operations are primitive and hardware optimized such as shift and logical operations.

The second reason is the need for a decoder that decompresses the image and decode it into its raw format before applying the encryption algorithm, as well as an encoder to compress it again into the original or another well-known image format. The same process should be repeated in the decryption phase too. These operations take time and their speeds depend on multiple parameters such as the original image format, resolution, compression ratio …etc. In papers suggesting new image encryption schemes, these extra steps of encoding and decoding are usually ignored. The performance analysis in the papers focuses usually on measuring the time taken by the pixel randomization process. Therefore, the performance analysis provided with such ciphers is usually irrelevant for real world scenarios.

Finally, there is another reason of the performance difference that can affect the results in this experiment. This reason is the use of a high level garbage-collected language like JavaScript for implementing the pixel-wise cipher. On the other hand, the AES implementation present in Node.js is the OpenSSL version which is highly optimized and implemented using C, a very performant low level compiled language. However, time is money and implementing the chaotic scheme in C takes more time and resources than doing it using JavaScript. We decided to do the JavaScript version as a prototype and planned to re-implement it later in C or C++ if it gets wins against AES in this test suite. However, it turns out that the cipher has many performance issues other than the programming language used for its implementation.

AES clearly wins the raw speed test.

4.6.7 Server Load Test

To see the potential impact of each cipher on the responsiveness of the application, we created a simple Node.js server that reads a previously encrypted image from the storage space, decrypts it and sends it in response. This server is then load-tested using a load test tool called "Autocannon". The load test is performed using all the 3 test images. Results are summarized in Table 10.

Table 10: Server load test results

	AES (Req/Sec)	**Chaotic (Req/Sec)**
Cameraman	1750.7	42.4
Baboon	1862.5	21.2
Minaret	108	0

The number of requests per second corresponding to each cipher was expected because it is directly related to the raw speed of the cipher. AES was able to handle requests up to 1.8K Req/Sec serving small images, while the chaotic cipher reaches ~43 Req/Sec at maximum and was not able to resist the test load decrypting the medium sized minaret image.

AES wins this server load test.

4.7 Conclusion

In this chapter, a comparative analysis was done between AES cipher and a recently proposed chaotic cipher in the context of selecting the right cipher to use in a web application.

According to the real needs of the application, seven tests have been designed to test the suitability of each cipher to the needs of the application.

We found that AES outperforms the chaotic cipher in approximately all tests. AES passes all the seven tests while the pixel-wise cipher passes only one test and failed in the rest.

Some issues with the studied chaotic cipher were found to be common to other pixel-wise ciphers. These issues are not related to a specific implementation but a flaw in the principle of pixels' encryption itself. Such as the leak of information, the symmetry and metadata loss issue as well as the significant increase in size of encrypted image and the degradation of performance by requiring encoders and decoders to deal with images on each encryption/decryption operation in real world applications.

This study proved that satisfying the well-known metrics in the field of image encryption field is not enough to have a production ready general purpose image cipher. The designers of such schemes need to take into account the different contexts and needs of real world applications.

Finally, the tests above are designed in the context of a specific web application. The final judgment was taken in this specific context and it might differ if the context changes. Depending on the level of security and performance needed and the level of tolerance toward the loss of metadata and data size increase, the studied chaotic cipher can be good enough for some other fields or applications with different needs and different requirements.

Chapter 5

A Highlight on Some Common Image Encryption Design Issues

Index

Chapter 5

A Highlight on Some Common Image Encryption Design Issues

5.1 Introduction

Based on our previous experience in chapter 4 trying to implement a chaotic encryption scheme into a real web application, a set of image encryption anti-patterns and issues were discovered. Most of those issues are affecting the well-known pixel-wise model itself not only specific algorithms. They are common across a large set of recently proposed schemes following that model regardless of the technique used within the encryption algorithm in most cases.

Not paying attention to these issues leads encryption schemes' designers to focus on optimizing the already optimized parts while ignoring some real issues that need to be addressed as early as possible.

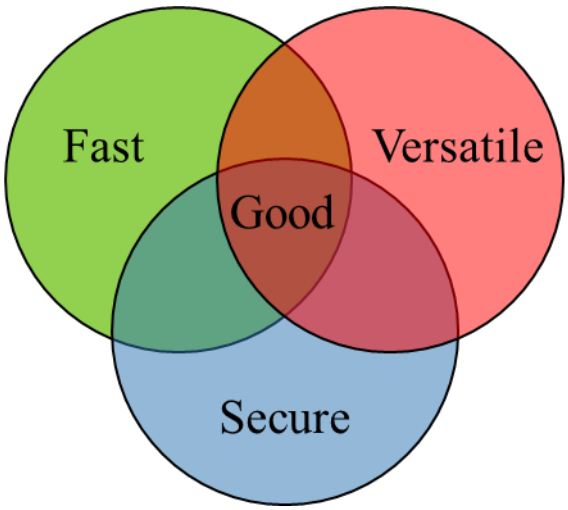


Figure 5-1: Characteristics of a good cipher

As illustrated in Figure 5-1, a good general purpose cipher is required to be fast, secure and usable with a large number of use cases. Slow algorithms are not suited for applications with large amounts of data to encrypt. Insecure ones make data vulnerable to attackers. While very specific algorithms cannot be used in the majority of real world applications and will miss the required inspection by cryptanalysts to assess their security and fix any potential vulnerabilities.

In this chapter we will summarize the issues and limitations that we've found through the analysis of the pixel-wise image encryption model. The results will be divided into 3 categories: Security, performance and usability.

5.2 Security Issues

Security issues are the most serious issues that can be found in a cryptosystem. The cryptosystem main purpose is to provide security via data encryption. Therefore, finding security issues in a given scheme defeats the whole purpose of encrypting data. The following sections will be a summary of the security issues found in that general image encryption model mentioned earlier.

5.2.1 Information Leakage

The purpose of encryption is to hide all information about the data being encrypted. Recent image encryption algorithms generally follow the pixel-wise model which operate on the pixel level of the image. Usually, they take the pixels of the plain image, scramble them and output a new image with random looking pixels.

Algorithms following this famous model can be robust enough to be irreversible by known attacks. However, they have some security issues common to all of them. Figure 5-2 illustrates the information that can be known by an attacker having only the encrypted image at his disposal. These pieces of information are:

- The type of the plain data, in this case it is obvious that the original data is an image.

- The dimensions and the size of the plain image.

- The number of channels of the plain image.

Whether the attacker can reveal the exact original content of the plain image or not, knowing only the above information about the plain data makes the cipher semantically unsecure. Semantic security insists on the fact that an attacker should know nothing about the plain data

and cannot distinguish the ciphertext from random. Depending on the application, the revealed information can be very dangerous to be known. It can act as a basis on which an attacker can build more advanced and targeted attacks.

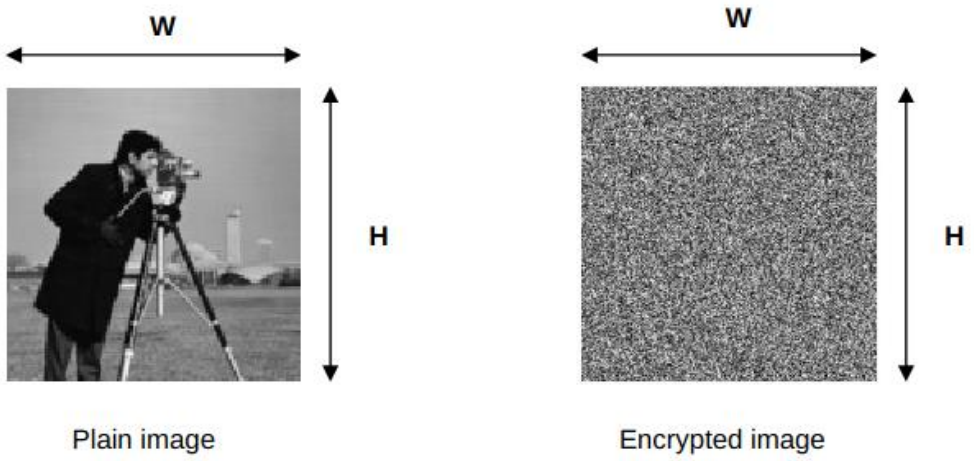


Figure 5-2: Information leakage

If such image is encrypted using a classical encryption method like AES for example, these known pieces of information will not be leaked. AES encrypts blocks of binary data. The resulting output will not reveal any information about the image dimensions or the even its data type.

5.2.2 Lack of data integrity

An encrypted piece of data is usually meant to be transferred from the sender to the receiver through a public channel, or to be stored in a public storage in order to be retrieved back later. In such environments we can distinguish two general types of attacks: passive attacks and active attacks. Passive attacks are attacks where the attacker tries to reveal the plain data or part of it only. On the other hands, active attacks are attacks that try to maliciously manipulate the encrypted data itself so the decrypted data will be different from the original one. If an attacker can modify the encrypted data without being detected, the active attack is considered successful.

Figure 5-3 illustrates how a Man in the middle attack can be performed on image encryption schemes that don't offer an integrity check. The cipher image is sent in a public channel. An attacker can modify it maliciously before it gets to the recipient. When the recipient decrypts

it, he gets a modified plain image. However, he is not going to know whether it has been modified or not. This is considered a successful active attack.

The more knowledge the attacker has about the plain image and encryption algorithm, the more dangerous the modification can be. Many implementations of the image encryption model in study are vulnerable to active attacks. Usually they have no integrity check mechanism to stand against this type of attacks.

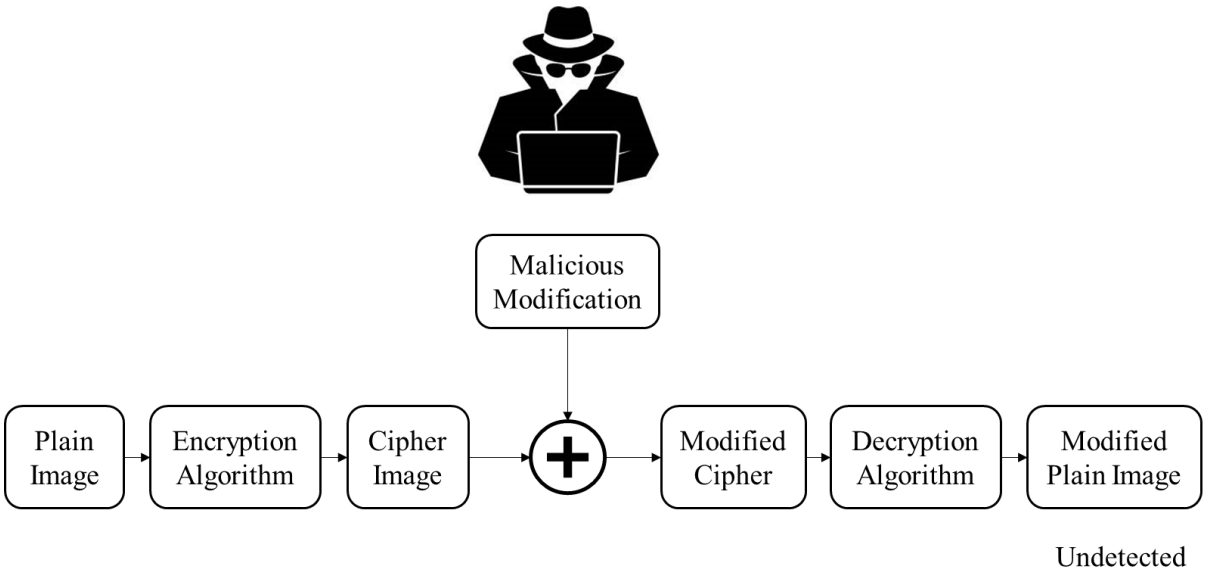


Figure 5-3: Man In the Middle Attack on encrypted image

Unlike many other implementations, in 2018, an encryption algorithm that combined Hill Cipher method, Morse Code and Least Significant Bit algorithm addressed the issue of data integrity [49]. However, many other new algorithms don't offer any way to preserve the integrity of encrypted images.

In classical cryptography, this issue was addressed by introducing the concept of "Authenticated Encryption", which is the combination of both encryption and authentication like GCM (Galois Counter Mode) mode for example. Using GCM mode, if a single bit of ciphertext is modified it will be detected in decryption.

5.2.3 Lack of Randomization

Papers proposing new image encryption algorithms tend to focus generally on the encryption of one separate image. They don't usually provide details on the use of the algorithm with a set of images. With the absence of such details, the intuitive approach to encrypt many images is

to feed each image into the encryption algorithm separately. The result will be a set of cipher images corresponding to a set of plain images.

An image encryption algorithm can be modelled as a block cipher with a variable block size depending on the image size being encrypted. The method described above of encrypting multiple images is called ECB (Electronic Code Book) mode of operation. ECB mode is the simplest and fastest method of using a block cipher to encrypt data larger than one block in size. However, ECB is found to be insecure [61].

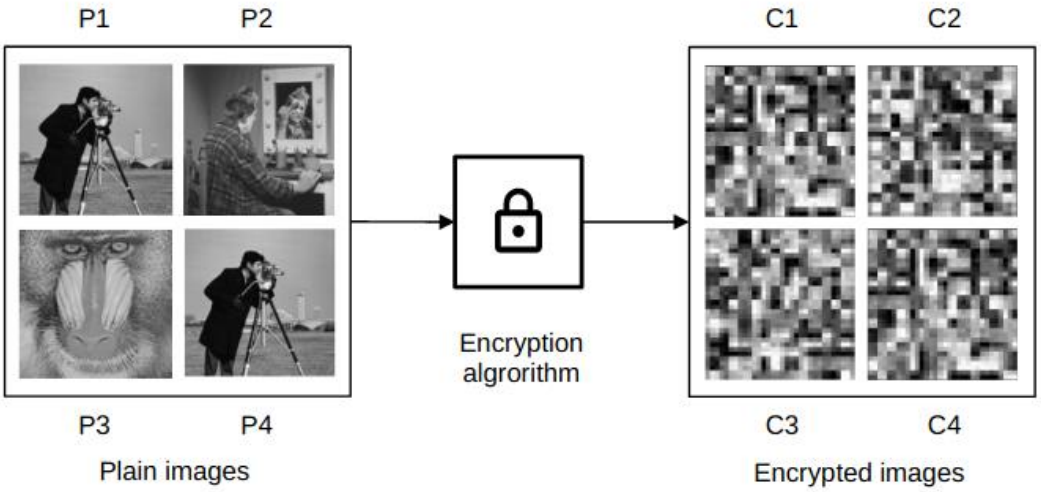


Figure 5-4: Revealing patterns when encrypting similar images

Figure 5-4 illustrates a real world demonstration on the vulnerability of using image encryption algorithms with an ECB-like mode of operation. Supposed there are multiple images inside a folder that are going to be encrypted. The result will be a folder with an equal number of encrypted images. If two plain images are identical, two cipher images are going to be identical too. Therefore, an attacker can get some information on the plain images only by looking to the encrypted data. Even if the cipher images are not distinguishable visually, there are computational methods to detect similarity such as calculating the correlation coefficient or the image hash value using one of the well-known hashing algorithm.

The way classic cryptography overcame this problem is by introducing various modes of operations (View Chapter 1). The goal of modes of operation is to insure that encrypting the same data twice doesn’t output the same result. Image encryption algorithms designers needs

to pay attention to developing techniques to deal with this situation too. Encrypting similar data is very common in real applications. Therefore, this issue should not be overlooked.

The encryption scheme studied in the previous chapter has successfully solved this problem by introducing 2 random rows on top and bottom of the image. These rows are then confused with the rest of the image are appended to the cipher image to be used in decryption phase [35]. Some other ciphers attempted to solve this problem too but resulted in the introduction of another kind of vulnerabilities as will be discussed in section 5.4.1 below.

5.3 Performance Issues

A general purpose cryptosystem is required to be fast and resource efficient. If an algorithm is secure but takes long time to execute, it won't be appropriate for applications where large amounts of data are processed. One common metric to measure an algorithm's performance is the big O notation, which is a metric that tells how the algorithm execution time increases according to the input size. Another common method is to compare the algorithm to similar algorithms using the same input like we did in chapter 4. In the following sections, some performance issues found in the image encryption model previously mentioned are presented.

5.3.1 Requirement of encoding and decoding

In real world, images are not stored as a matrix of pixels. Instead, they are compressed and encoded into some well-known image formats like: JPEG, PNG, GIF, WEBP ...etc. On the other hand, the pixel-wise encryption model implemented by many algorithms recently operates on a matrix of pixels. Therefore, there is an implicit need for a decoder that decompresses the image and decode it into pixels before applying the encryption algorithm. As well as an encoder to compress it again into the original or another well-known image format. The same process in should be repeated in the decryption phase again.

Figure 5-5 illustrates the hidden extra steps required before and after applying the image encryption scheme on real images. Since encryption algorithm is applied on image pixels and not on binary image files directly, encoding and decoding steps are required and cannot be skipped.

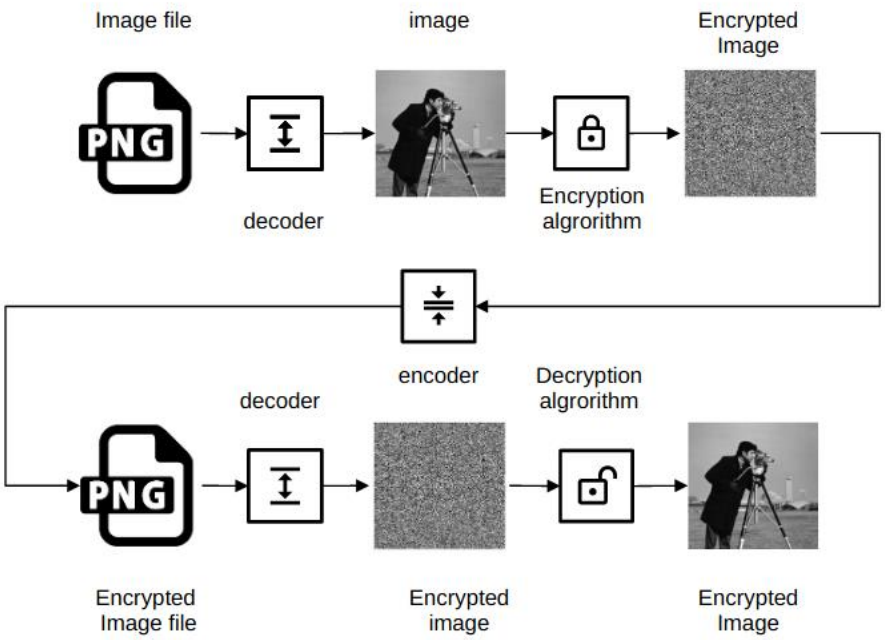


Figure 5-5: The need for encoders and decoders when using the encryption scheme

If such algorithm is used in a real application to encrypt an image file, the total encryption time will be equal to the sum of decoding, encryption and encoding times as in Eq30.

$$totalTime = decodingTime + encryptionTime + encodingTime \qquad Eq\ 30$$

On the other side decryption time will be calculated the same way. Encoding and decoding should be taken into account when decrypting an image too.

Using a classical general purpose encryption algorithm like AES for example to encrypt an image doesn't require any of these extra steps. AES encrypts the binary data directly and doesn't need neither encoding nor decoding. Comparing AES performance to these image encryption algorithms needs to be done against the whole process including encoding and decoding, comparing it with pixels' randomization phase only is misleading [62].

Ignoring these extra steps of computation makes the performance analysis provided in some papers irrelevant and unrealistic. The actual time for real usage will always be bigger than the time provided in the papers. These hidden extra steps should be taken into consideration when analyzing the performance of the algorithms for a more accurate assessment.

5.3.2 Inefficiency with compressed images

As said in the previous section, images in real world are usually stored and transmitted in a compressed format to save disk space and bandwidth. Therefore, a decoder and an encoder before and after the encryption algorithm are required. In addition to the performance issue discussed earlier, there is another issue with such design, which is that the encryption process will most likely increase the size of the provided image.

In order to encrypt a specific image, the image file needs to be decoded first, encrypted then encoded back to either the original or another format. The aim of image encryption is to randomize the pixel values of the image. Therefore, the more random are the pixels the better the encryption algorithm will be. However, this pixel randomization breaks the correlation between image pixels and eliminates similar patterns. Hence, it reduces the effect of compression on the encrypted image. As a result, the encrypted image size will most likely be bigger than the plain image one. This increase in size will have undesirable consequences on the storage and transmission performance.

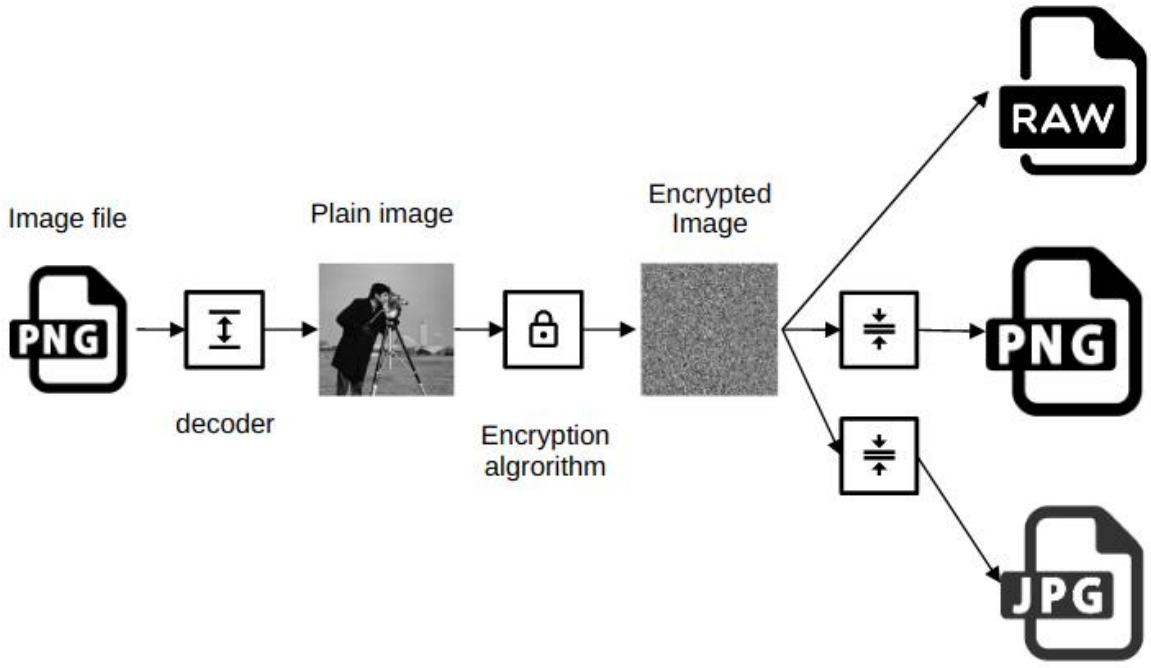


Figure 5-6: The possible types of image encryption outputs

Figure 5-6 shows the 3 possible options of the encryption output image: uncompressed image, lossless compressed image and lossy compressed image. All of these options have undesired effects.

Outputting the encrypted image uncompressed preserve all data but it is the worst option from a file size perspective. The size of an uncompressed image will be larger than all the other options.

Lossless compression also preserves all data. However, the lossless compression algorithm will have negligible effects since the pixels are randomized and the patterns are broken. The resulting file size will be approximately equal to the size of an uncompressed image.

The 3rd option of lossy-compress the encrypted image might reduce the file size a little bit. However, it leads to losing part of the image data in compression. Therefore, the decryption process will not produce the exact plain image. Losing data is not an accepted compromise for good a cipher.

5.4 Usability Issues and Limitations

Security and performance are not the only requirements for a good cipher. A cryptosystem needs to be usable and versatile. Next are some usability issues found in the model in study.

5.4.1 Randomness by dependency on plain image

In this chapter, in section 5.2.3 we discussed the lack of randomness issue when encrypting the same image twice that is common to many algorithms. Some algorithms tried to address this issue by introducing a dependency on the plain image itself. The idea is to get a new value that is a function of the plain image (ex: pixels' average, a pixel at a given position ...etc.). This value is then used to alter the encryption key. Since the encryption process is too sensitive to the key, a slight change in the image will produce a change in the key which produce a completely different output image. The idea of plain image dependency on encryption works fine in encryption phase. However, it fails in decryption either by introducing a security vulnerability or a usability issue.

In [57] a new image encryption algorithm with dependency on plain image was proposed. The dependency was introduced by calculating the image pixels' average and use it to modify the initial conditions. A similar method was used in [33], where a random pixel value was selected from the permuted image and used as part of the encryption key. Both methods used that plain image dependent value in the decryption algorithm without mentioning how that value has been transmitted to the decryption phase. There exist two possible options, either it is transmitted publicly or securely as illustrated in Figure 5-8 and both approaches have flaws.

If the value is transmitted publicly, it will introduce a security vulnerability. The algorithm will not be semantically secure. An attacker will know some information about the plain image, being either the pixel average or any other function of the plain image. Knowing an information like that about the plain image can reduce the scope of brute-force attacks.

Otherwise, requiring a secure transmission of that info is a usability issue. If a reliable and secure transmission channel exists, it is better and more resource efficient to transmit images in plain without encryption in that channel. Encryption will not be needed in such conditions. It will be just a waste of time and resources.

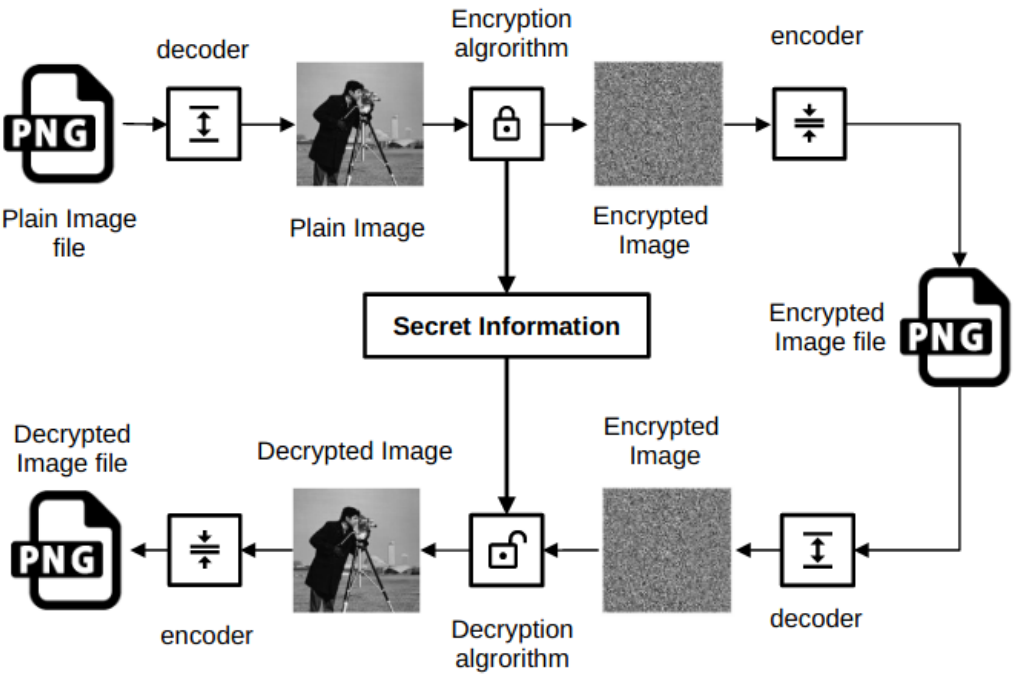


Figure 5-7: Principle of algorithms requiring the retain of a secret info

Some other researchers proposed another method where a plain image dependency is introduced in encryption, but instead of transmitting it to the decryption algorithm, it is recalculated again from the encrypted image itself [58]. However, this method has a serious security vulnerability. In order to avoid sending the calculated value to the decryption algorithm, one quarter of the image was not confused with the calculated key. Having the encrypted image only, one quarter of the plain image can be restored in its permuted form by dividing the image into 4 blocks and XORing the 3rd block with the 4th one which is a serious security issue.

5.4.2 Ignoring image metadata

When an image is taken with a digital camera, the digital camera usually stores some text information alongside the image called metadata, like the location, the capture time, the camera settings … etc. An image encryption algorithm should be able to encrypt these metadata and decrypt it correctly. It should neither discard it nor output it in plain format.

Figure 5-8 represents some metadata tags of a real image taken with a digital camera Canon EOS Kiss X4. The metadata was viewed using GIMP the free and open source image editing software. The tags visible in this screenshot is just a fraction of all the tags available on the image.

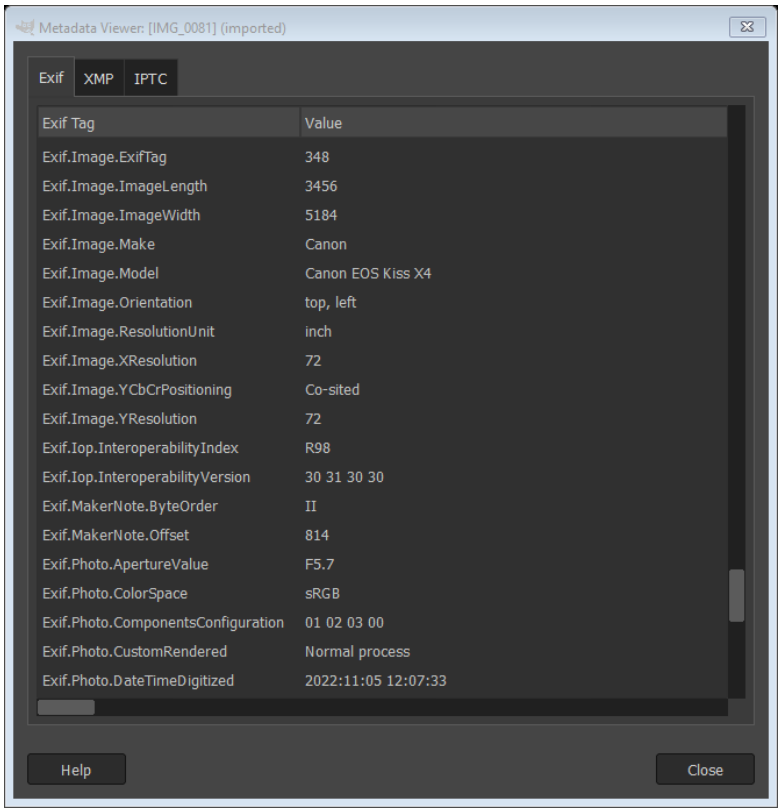


Figure 5-8: Example of real image metadata

Many new image encryption algorithms focus on image pixels and ignore this important piece of information and don't offer details on how to handle it. In consequence, if an engineer wants to implement such algorithms in a real system, he will be faced with 3 options: ignoring metadata, outputting it in plain format or using another algorithm like AES to encrypt it and inject it back into the image as illustrated in Figure 5-9.

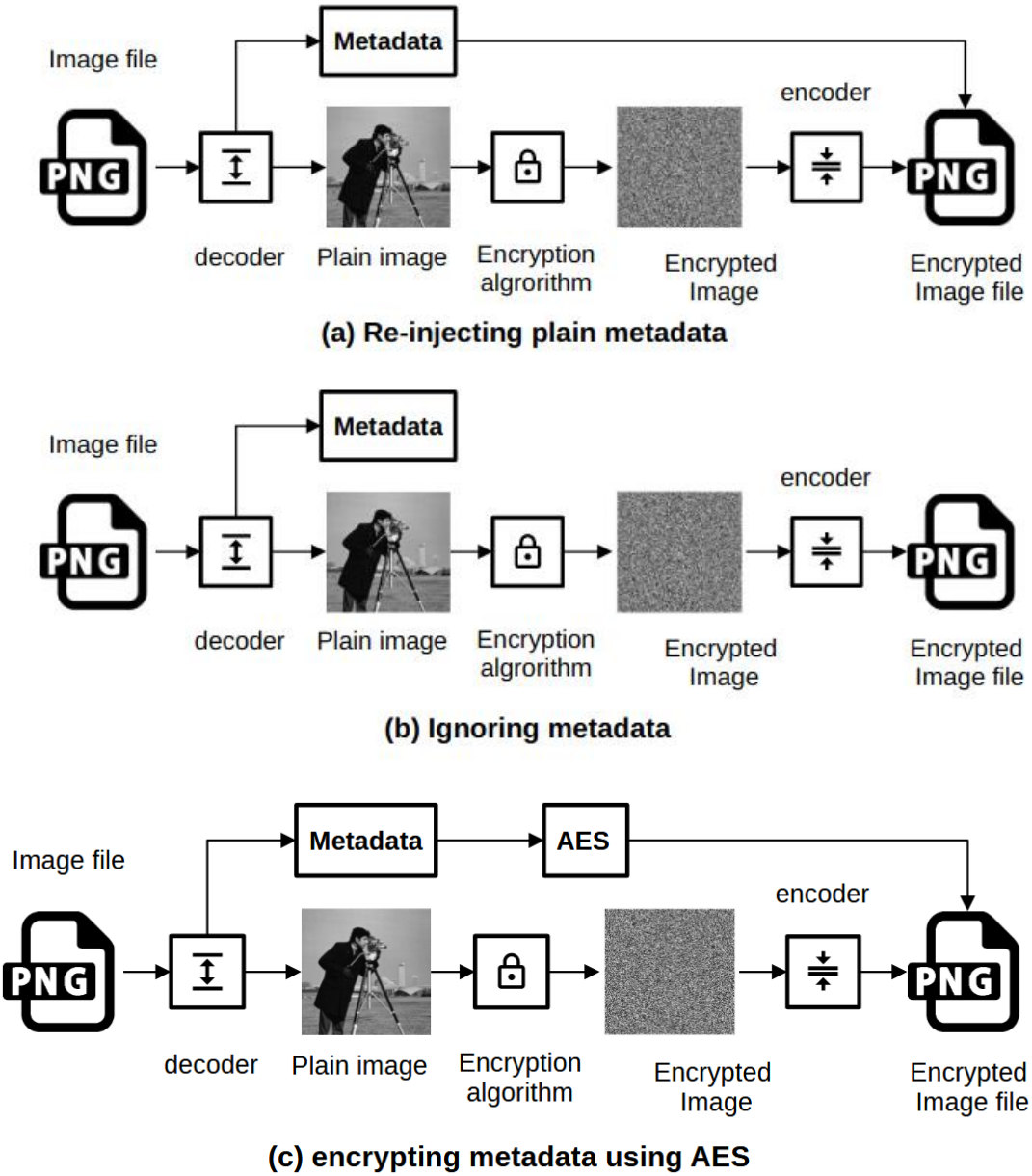


Figure 5-9: Available options in handling image metadata

A symmetric cryptosystem by definition should return the same plain data after encryption and decryption using the same key. It should neither add additional data nor cause a data loss. Ignoring a piece of information is a serious problem that could not be acceptable in a good cryptosystem. Therefore, ignoring metadata is not a valid option.

On the other hand, outputting metadata in plain is a big security issue. This defeats the purpose of encryption at all since an important part of the plain data is known from the ciphertext. This is not considered a good option too.

Finally, using another algorithm like AES to encrypt metadata can solve the problem of security and symmetry. However, it introduces another usability problem. It raises the question about the purpose of the image encryption algorithm. If AES is going to be used anyway to encrypt the image metadata, why not use it to encrypt the image itself too? It is hard to imagine how this hybrid and complex approach will be better than using a standard algorithm to encrypt the whole data.

Image metadata is an important piece of information usually associated with the majority of digital images. New image encryption schemes should take that information into consideration in order to be usable in real world applications.

5.5 Conclusion

In this chapter, some common issues in the pixel-wise image encryption algorithms were highlighted. The focus was on the most common issues repeated by many designers over the years. Most of those issues show up when trying to implement the algorithms into real world applications. Actually, good ciphers are designed to secure real data and to be used by real people in real and very sensitive applications. Therefore, ciphers need to be designed carefully with attention to those real world problems. They need to be optimized with the end user experience and security in mind. Satisfying the well-known metrics is not enough to have a good cryptosystem. Ciphers should be secure, fast, and usable in real engineering scenarios and use cases.

General Conclusion and Perspectives

The work done is this thesis was a contribution to the information security domain. A novel hybrid method of cryptanalysis was proposed to break a recently published image encryption algorithm. This hybrid method was based on a series of chosen-plaintext attacks and a brute-force attack. The encryption scheme was totally cracked by the cryptanalysis algorithm that we proposed.

During the time that we spent working on this thesis, we were curious to see how these algorithms are being used in real engineering applications and communication mediums. However, we were not able to find real applications using these kind of algorithms. That led us to trying to implement a recently proposed chaos based algorithm claimed to be fast and secure in a real world image sharing web application. After designing a custom test suite based on real engineering requirements and comparing that chaotic scheme to AES, we found that AES outperformed it in the majority of tests. In some tests like speed and server load, AES was orders of magnitude faster and capable of handling incoming requests.

After the researches that we did about classic cryptosystems and image encryption schemes, and with the help of our comparative analysis conducted in the fourth chapter, we detected some common anti-patterns repeatedly committed among many image encryption algorithms recently proposed. We collected those issues and highlighted them in the fifth chapter of this thesis. The goal is to draw the attention of cryptosystems' designers to those common issues and vulnerabilities in the future. We expect this to save some research time and effort in designing future systems.

Coming from a software engineering background, and after the comparative analysis we have performed and the common issues found in the pixel-wise encryption model, we found ourselves questioning the practicality of the whole idea of developing encryption algorithms operating on the image pixels' level. They may have their small set of applications where they perform well. However, due to the serious issues they have which have been detected in chapter 4 and highlighted in more details in chapter 5, we cannot be too optimistic about them getting a wide adoption in real world applications in the near future.

In the journey of working on this thesis, we saw a trend of constantly developing new image encryption algorithms. Each year, hundreds of image ciphers are published. However, many of them have the same issues we discussed earlier, and we saw in chapter 4 how using standard

cryptosystems like AES for example is more performant, practical and secure in many aspects. Therefore, the added value of those research efforts is very minimal.

It is hard to think about any real system where a pixel-wise image-only encryption algorithm can be useful and more practical than the standard encryption algorithms. Real systems tend to have a variety of data types. Therefore, data type agnostic ciphers are preferred. Anyway, we can be proved wrong in the future and such system may exist where image-only algorithms are superior. However, we think that the existence of this system should precede the development of its specific algorithms. We believe that developing solutions should be driven by the necessity for them and not the inverse.

Our conclusion is that encryption algorithms specific to images or like we have called them throughout this thesis: pixel-wise encryption algorithms, are not superior than standard ciphers like AES in most of real applications. In addition, it is hard to find real systems where this kind of algorithms makes sense due to the flaws by design that it has that are not generally tolerated in real world. Cryptosystems needs to be developed with real applications and the end users in mind.

At the end, a good perspective we see for future works in this field is trying to address the existing flaws presented in this thesis. We hope to see future researches that continue developing image encryption algorithms with practical and real use cases in mind.

References

[1] J. von zur Gathen, *CryptoSchool*. Berlin, Heidelberg: Springer, 2015. doi: 10.1007/978-3-662-48425-8.

[2] H. Delfs and H. Knebl, *Introduction to Cryptography: Principles and Applications*, Softcover reprint of the original 3rd ed. 2015 edition. Place of publication not identified: Springer, 2016.

[3] C. E. Shannon, "A mathematical theory of cryptography. Mathematical Theory of Cryptography," 1945.

[4] "Understanding Cryptography: A Textbook for Students and Practitioners: Paar, Christof, Pelzl, Jan, Preneel, Bart: 9783642446498: Books - Amazon." Accessed: May 30, 2022. [Online]. Available: https://www.amazon.com/Understanding-Cryptography-Textbook-Students-Practitioners/dp/3642446493

[5] W. Diffie and M. E. Hellman, "New Directions in Cryptography," *IEEE Transactions on Information Theory*, p. 1976.

[6] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 120–126, Feb. 1978, doi: 10.1145/359340.359342.

[7] N. P. Smart, *Cryptography: An Introduction*. McGraw-Hill, 2003.

[8] C. E. Shannon, "Communication Theory of Secrecy Systems," 1949.

[9] J. Daemen and V. Rijmen, *The design of Rijndael*, 2nd ed. Springer, 2002.

[10] D. Liu *et al.*, Eds., "Chapter 3 - An Introduction To Cryptography," in *Next Generation SSH2 Implementation*, Burlington: Syngress, 2009, pp. 41–64. doi: 10.1016/B978-1-59749-283-6.00003-9.

[11] T. Popp, *An introduction to implementation attacks and countermeasures*. 2009, p. 115. doi: 10.1109/MEMCOD.2009.5185386.

[12] F. E. A. El-Samie *et al.*, *Image Encryption: A Communication Perspective*, 1st edition. CRC Press, 2017.

[13] "Image file formats - Easy explained | Adobe." Accessed: Dec. 05, 2022. [Online]. Available: https://www.adobe.com/creativecloud/file-types/image.html

[14] J. Miano, *Compressed Image File Formats: JPEG, PNG, GIF, XBM, BMP*. Addison-Wesley Professional, 1999.

[15] "What are PNG files and how do you open them? | Adobe." Accessed: Dec. 05, 2022. [Online]. Available: https://www.adobe.com/creativecloud/file-types/image/raster/png-file.html

[16] G. K. Wallace, "The JPEG still picture compression standard," *IEEE Trans. Consum. Electron.*, vol. 38, no. 1, pp. xviii–xxxiv, Feb. 1992, doi: 10.1109/30.125072.

[17] W. Song, C. Fu, M. Tie, C.-W. Sham, J. Liu, and H. Ma, "A fast parallel batch image encryption algorithm using intrinsic properties of chaos," *Signal Process. Image Commun.*, vol. 102, p. 116628, Mar. 2022, doi: 10.1016/j.image.2021.116628.

[18] I. Ahmad and S. Shin, "A novel hybrid image encryption–compression scheme by combining chaos theory and number theory," *Signal Process. Image Commun.*, vol. 98, p. 116418, Oct. 2021, doi: 10.1016/j.image.2021.116418.

[19] Z. Man, J. Li, X. Di, Y. Sheng, and Z. Liu, "Double image encryption algorithm based on neural network and chaos," *Chaos Solitons Fractals*, vol. 152, p. 111318, Nov. 2021, doi: 10.1016/j.chaos.2021.111318.

[20] W. Dong, Q. Li, and Y. Tang, "Image encryption-then-transmission combining random sub-block scrambling and loop DNA algorithm in an optical chaotic system," *Chaos Solitons Fractals*, vol. 153, p. 111539, Dec. 2021, doi: 10.1016/j.chaos.2021.111539.

[21] S. Wang, Q. Peng, and B. Du, “Chaotic color image encryption based on 4D chaotic maps and DNA sequence,” *Opt. Laser Technol.*, vol. 148, p. 107753, Apr. 2022, doi: 10.1016/j.optlastec.2021.107753.
[22] M. Yildirim, “Optical color image encryption scheme with a novel DNA encoding algorithm based on a chaotic circuit,” *Chaos Solitons Fractals*, p. 111631, Nov. 2021, doi: 10.1016/j.chaos.2021.111631.
[23] U. Hayat and N. A. Azam, “A novel image encryption scheme based on an elliptic curve,” *Signal Process.*, vol. 155, pp. 391–402, Feb. 2019, doi: 10.1016/j.sigpro.2018.10.011.
[24] N. Sasikaladevi, K. Geetha, K. Sriharshini, and M. Durga Aruna, “H3-hybrid multilayered hyper chaotic hyper elliptic curve based image encryption system,” *Opt. Laser Technol.*, vol. 127, p. 106173, Jul. 2020, doi: 10.1016/j.optlastec.2020.106173.
[25] A. B. Joshi, D. Kumar, A. Gaffar, and D. C. Mishra, “Triple color image encryption based on 2D multiple parameter fractional discrete Fourier transform and 3D Arnold transform,” *Opt. Lasers Eng.*, vol. 133, p. 106139, Oct. 2020, doi: 10.1016/j.optlaseng.2020.106139.
[26] C. Li and X. Yang, “An image encryption algorithm based on discrete fractional wavelet transform and quantum chaos,” *Optik*, p. 169042, Apr. 2022, doi: 10.1016/j.ijleo.2022.169042.
[27] Z. Li, H. Zhang, X. Liu, C. Wang, and X. Wang, “Blind and safety-enhanced dual watermarking algorithm with chaotic system encryption based on RHFM and DWT-DCT,” *Digit. Signal Process.*, vol. 115, p. 103062, Aug. 2021, doi: 10.1016/j.dsp.2021.103062.
[28] X. Wang, C. Liu, and D. Jiang, “A novel triple-image encryption and hiding algorithm based on chaos, compressive sensing and 3D DCT,” *Inf. Sci.*, vol. 574, pp. 505–527, Oct. 2021, doi: 10.1016/j.ins.2021.06.032.
[29] T. Bekkouche, “Développement et implémentation des techniques de cryptage des données basées sur les transformées discrètes,” Thesis, 2018. Accessed: Jun. 05, 2022. [Online]. Available: http://dspace.univ-setif.dz:8888/jspui/handle/123456789/2776
[30] M. Kaur and V. Kumar, “A Comprehensive Review on Image Encryption Techniques,” *Arch. Comput. Methods Eng.*, vol. 27, no. 1, pp. 15–43, Jan. 2020, doi: 10.1007/s11831-018-9298-8.
[31] H. M. Elkamchouchi and M. A. Makar, “Measuring encryption quality for bitmap images encrypted with rijndael and KAMKAR block ciphers,” in *Proceedings of the Twenty-Second National Radio Science Conference, 2005. NRSC 2005.*, Mar. 2005, pp. 277–284. doi: 10.1109/NRSC.2005.194011.
[32] D. E. Newton, *Encyclopedia of Cryptology*, Illustrated edition. Santa Barbara, Calif: ABC-CLIO, 1997.
[33] S. Anwar and S. Meghana, “A pixel permutation based image encryption technique using chaotic map,” *Multimed. Tools Appl.*, vol. 78, no. 19, pp. 27569–27590, Oct. 2019, doi: 10.1007/s11042-019-07852-2.
[34] A. Mokhnache and L. Ziet, “Cryptanalysis of a Pixel Permutation Based Image Encryption Technique Using Chaotic Map,” *Trait. Signal*, vol. 37, no. 1, pp. 95–100, Feb. 2020, doi: 10.18280/ts.370112.
[35] M. Z. Talhaoui, X. Wang, and M. A. Midoun, “Fast image encryption algorithm with high security level using the Bülban chaotic map,” *J. Real-Time Image Process.*, vol. 18, no. 1, pp. 85–98, Feb. 2021, doi: 10.1007/s11554-020-00948-1.
[36] O. Alpar, “Analysis of a new simple one dimensional chaotic map,” *Nonlinear Dyn.*, vol. 78, no. 2, pp. 771–778, Oct. 2014, doi: 10.1007/s11071-014-1475-1.
[37] O. S. Faragallah *et al.*, “Efficient optical double image cryptosystem using chaotic mapping-based Fresnel transform,” *Opt. Quantum Electron.*, vol. 53, no. 6, p. 305, Jun. 2021, doi: 10.1007/s11082-021-02864-5.

[38] R. Ni, F. Wang, J. Wang, and Y. Hu, “Multi-Image Encryption Based on Compressed Sensing and Deep Learning in Optical Gyrator Domain,” *IEEE Photonics J.*, vol. 13, no. 3, pp. 1–16, Jun. 2021, doi: 10.1109/JPHOT.2021.3076480.
[39] B. Bouteghrine, C. Tanougast, and S. Sadoudi, “Novel image encryption algorithm based on new 3-d chaos map,” *Multimed. Tools Appl.*, vol. 80, no. 17, pp. 25583–25605, Jul. 2021, doi: 10.1007/s11042-021-10773-8.
[40] F. Musanna, D. Dangwal, and S. Kumar, “Novel image encryption algorithm using fractional chaos and cellular neural network,” *J. Ambient Intell. Humaniz. Comput.*, vol. 13, no. 4, pp. 2205–2226, Apr. 2022, doi: 10.1007/s12652-021-02982-8.
[41] Z. Man *et al.*, “A novel image encryption algorithm based on least squares generative adversarial network random number generator,” *Multimed. Tools Appl.*, vol. 80, no. 18, pp. 27445–27469, Jul. 2021, doi: 10.1007/s11042-021-10979-w.
[42] S. Zhou, X. Wang, Y. Zhang, B. Ge, M. Wang, and S. Gao, “A novel image encryption cryptosystem based on true random numbers and chaotic systems,” *Multimed. Syst.*, vol. 28, no. 1, pp. 95–112, Feb. 2022, doi: 10.1007/s00530-021-00803-8.
[43] G. Ye, M. Liu, and M. Wu, “Double image encryption algorithm based on compressive sensing and elliptic curve,” *Alex. Eng. J.*, vol. 61, no. 9, pp. 6785–6795, Sep. 2022, doi: 10.1016/j.aej.2021.12.023.
[44] J. Du, Y. Xiong, C. Wu, and C. Quan, “Optical image encryption with high efficiency based on variable-distance ghost imaging,” *Optik*, vol. 252, p. 168484, Feb. 2022, doi: 10.1016/j.ijleo.2021.168484.
[45] S. Anwar and S. Meghana, “A pixel permutation based image encryption technique using chaotic map,” *Multimed. Tools Appl.*, vol. 78, no. 19, pp. 27569–27590, Oct. 2019, doi: 10.1007/s11042-019-07852-2.
[46] P. Refregier and B. Javidi, “Optical image encryption based on input plane and Fourier plane random encoding,” *Opt. Lett.*, vol. 20, no. 7, pp. 767–769, Apr. 1995, doi: 10.1364/OL.20.000767.
[47] R. Tao, J. Lang, and Y. Wang, “Optical image encryption based on the multiple-parameter fractional Fourier transform,” *Opt. Lett.*, vol. 33, no. 6, pp. 581–583, Mar. 2008, doi: 10.1364/OL.33.000581.
[48] Y. Zhang, L. Zhang, Z. Zhong, L. Yu, M. Shan, and Y. Zhao, “Hyperchaotic image encryption using phase-truncated fractional Fourier transform and DNA-level operation,” *Opt. Lasers Eng.*, vol. 143, p. 106626, Aug. 2021, doi: 10.1016/j.optlaseng.2021.106626.
[49] D. Nofriansyah *et al.*, “A New Image Encryption Technique Combining Hill Cipher Method, Morse Code and Least Significant Bit Algorithm,” *J. Phys. Conf. Ser.*, vol. 954, p. 012003, Jan. 2018, doi: 10.1088/1742-6596/954/1/012003.
[50] J. Yu, W. Xie, Z. Zhong, and H. Wang, “Image encryption algorithm based on hyperchaotic system and a new DNA sequence operation,” *Chaos Solitons Fractals*, vol. 162, p. 112456, Sep. 2022, doi: 10.1016/j.chaos.2022.112456.
[51] X. Li, J. Zeng, Q. Ding, and C. Fan, “A Novel Color Image Encryption Algorithm Based on 5-D Hyperchaotic System and DNA Sequence,” *Entropy*, vol. 24, no. 9, Art. no. 9, Sep. 2022, doi: 10.3390/e24091270.
[52] L. Teng, X. Wang, and Y. Xian, “Image encryption algorithm based on a 2D-CLSS hyperchaotic map using simultaneous permutation and diffusion,” *Inf. Sci.*, vol. 605, pp. 71–85, Aug. 2022, doi: 10.1016/j.ins.2022.05.032.
[53] N. Tsafack, J. Kengne, B. Abd-El-Atty, A. M. Iliyasu, K. Hirota, and A. A. Abd EL-Latif, “Design and implementation of a simple dynamical 4-D chaotic circuit with applications in image encryption,” *Inf. Sci.*, vol. 515, pp. 191–217, Apr. 2020, doi: 10.1016/j.ins.2019.10.070.

[54] H. Liu, J. Liu, and C. Ma, “Constructing dynamic strong S-Box using 3D chaotic map and application to image encryption,” *Multimed. Tools Appl.*, Feb. 2022, doi: 10.1007/s11042-022-12069-x.

[55] X. Gao, J. Mou, S. Banerjee, Y. Cao, L. Xiong, and X. Chen, “An effective multiple-image encryption algorithm based on 3D cube and hyperchaotic map,” *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 34, no. 4, pp. 1535–1551, Apr. 2022, doi: 10.1016/j.jksuci.2022.01.017.

[56] D. Stuttard, “Security & obscurity,” *Netw. Secur.*, vol. 2005, no. 7, pp. 10–12, Jul. 2005, doi: 10.1016/S1353-4858(05)70259-2.

[57] T. Bekkouche, N. Diffellah, and S. Mokhnache, *Simple and Efficient Image Encryption Scheme based on Recursive Property and Plain Image-Chaotic Map Dependency*. 2019.

[58] X. Huang and G. Ye, “An efficient self-adaptive model for chaotic image encryption algorithm,” *Commun. Nonlinear Sci. Numer. Simul.*, vol. 19, no. 12, pp. 4094–4104, Dec. 2014, doi: 10.1016/j.cnsns.2014.04.012.

[59] W. Song, C. Fu, M. Tie, C.-W. Sham, J. Liu, and H. Ma, “A fast parallel batch image encryption algorithm using intrinsic properties of chaos,” *Signal Process. Image Commun.*, vol. 102, p. 116628, Mar. 2022, doi: 10.1016/j.image.2021.116628.

[60] L. Zhang, X. Liao, and X. Wang, “An image encryption approach based on chaotic maps,” *Chaos Solitons Fractals*, vol. 24, no. 3, pp. 759–765, May 2005, doi: 10.1016/j.chaos.2004.09.035.

[61] M. Vaidehi and B. J. Rabi, “Design and analysis of AES-CBC mode for high security applications,” in *Second International Conference on Current Trends In Engineering and Technology - ICCTET 2014*, Jul. 2014, pp. 499–502. doi: 10.1109/ICCTET.2014.6966347.

[62] A. Arab, M. J. Rostami, and B. Ghavami, “An image encryption method based on chaos system and AES algorithm,” *J. Supercomput.*, vol. 75, no. 10, pp. 6663–6682, Oct. 2019, doi: 10.1007/s11227-019-02878-7.