Higher Education and Scientific Research

University of Babylon

College of Science for Women

Computer Science Department

Image Encryption using Speck Lightweight Cryptography

Research submitted by student:

Hanen Taqi Abdoun

to the University of Babylon / College of Science for Women / Computer Science Department as part of the requirements for obtaining a Bachelor's degree in Computer Science

Supervised by:

Asst. Prof. PhD. Ahmed Badri Muslim

2024 A. D

بسم الله الرحمن الرحيم

وعَسَى أَنْ تَكْرَهُوا شَيْئًا وَهُوَ خَيْرٌ لَكُمْ وَعَسَى أَنْ تُحِبُّوا شَيْئًا وَهُوَ شَرٌّ لَكُمْ وَاللَّهُ يَعْلَمُ وَأَنْتُمْ لَا تَعْلَمُونَ

صدق الله العلي العظيم {البقرة}

الاهداء والشكر

الى كل من ساعدني وساهم في هذه البحث وبشكل خاص الدكتور احمد بدري مسلم اهدي لكم بحث تخرجي

لكل لحظة قضيتها في جهود البحث، لكل جهد مضنٍ وسهر ليالٍ، لكل تفانٍ واجتهاد، أهديكم هذا الشكر الصادق والامتنان العميق. لم يكن مجرد مسعى أكاديمي، بل كان رحلة استكشافية شيقة نحو العلم والمعرفة.

الاستاذ العزيز احمد بدري

أشكرك على تفانيك في كل خطوة من خطوات هذا المشوار، على صبرك وتحملك في وجه التحديات، وعلى إبداعك في تقديم الأفكار والنتائج بطريقة مبتكرة ومميزة . بفضلك وبفضل جهودك الجبارة، أصبح لدينا جزء جديد من المعرفة لنضيء بها طريقنا نحو المستقبل. فشكرًا لك من القلب على عطائك وإسهامك الثمين في إثراء العلم والمعرفة..

شكرًا للثقة التي وضعتها فيَّ ولإيمانك الراسخ بقدراتي. كانت ثقتك دافعًا قويًا بالنسبة لي للسعي نحو التميز والاختلاف في بحثي. إرشادك نحو التفكير النقدي والتحليلي، وتشجيعك المستمر، كانت عوامل رئيسية في نجاحي وتميزي.

أشعر أيضًا بالامتنان العميق تجاه نفسي، حيث تلقيت التعليم والدعم الذي ساهم في تطوير مهاراتي ونضجي كباحث. بفضل جهودي وإرشاداتك القيمة، تمكنت من تحقيق هذه البلوغة

أتمنى لك دوام النجاح والتألق في مسيرتك الأكاديمية والمهنية، وأن تستمتع بثمار عملك الجادة في كل مرحلة من مراحل حياتك

آمل أن يكون بحثنا التعاوني مساهمة قيمة في مجالنا، وأن يكون له تأثير إيجابي يمتد إلى الأجيال القادمة.

مع التقدير والاحترام الصادق

Table of contents

Abstract

In this project, the Speck algorithm was used to encrypt and decrypt grayscale images. Encryption was performed on three images of different sizes, which were then studied and analyzed. In term of performance, the execution time and throughput were computed for each image instance. Further, the histogram of each encrypted image and its corresponding original image was also measured. The histogram analysis shown that its very closed to the uniform distribution. The results obtained indicate that the Speck encryption algorithm has efficient performance to encrypt /decrypt the images of different size with very little difference in execution time.

Chapter one

1.1 Introduction:

In the contemporary landscape of digital security, the protection of sensitive information has emerged as a paramount concern. With the proliferation of digital data, including multimedia content like images, the need for robust encryption techniques capable of handling diverse data formats has become increasingly urgent. This project represents a significant endeavor aimed at addressing this need through the development of an innovative image encryption system utilizing the Speck algorithm.

The significance of this project lies in its dual focus on lightweight encryption algorithms and image encryption using Speck. Lightweight encryption algorithms are gaining prominence due to their efficiency and suitability for resource-constrained environments such as IoT devices and embedded systems. Speck, in particular, has garnered attention for its impressive performance and strong security properties, making it an ideal candidate for various encryption applications.

Furthermore, the encryption of images holds particular importance in today's digital landscape. Images often contain sensitive or confidential information, ranging from personal photographs to critical data in fields like healthcare and surveillance. Therefore, developing a secure and efficient image encryption solution is essential to safeguarding digital assets and ensuring privacy and confidentiality in various domains.

This project encompasses the development of an image encryption system that leverages the lightweight properties of the Speck algorithm. The system will be capable of encrypting images of various sizes while maintaining a balance between security, efficiency, and usability. By adapting Speck for image encryption and optimizing its performance, the project aims to provide a practical solution that meets the growing demand for secure image encryption techniques in today's digital landscape.

In the current digital era, the importance of lightweight encryption algorithms cannot be overstated. With the proliferation of connected devices and the Internet of Things (IoT), there is a growing need for encryption techniques that can operate efficiently on resource-constrained platforms. Lightweight algorithms offer a compelling solution to this challenge by providing robust security without imposing excessive computational overhead.

Among lightweight encryption algorithms, Speck has emerged as a standout candidate due to its impressive performance characteristics. Originally designed by the United States National Security Agency (NSA), Speck offers a compelling balance between security, speed, and simplicity, making it well-suited for a wide range of applications. Its efficient hardware implementation and strong cryptographic properties have led to its adoption in various domains, including telecommunications, IoT devices, and now, image encryption.

1.2 Project outline

- Chapter 1 describes the introduction of projects.
- Chapter 2 demo swats the thrutched background about cryptography at lightweight encryption methods.
- Chapter 3 describes the lightweight speck cryptography
- Chapter 4 conclusion the project

Chapter two
Encryption Systems

2.1 Introduction

In an era of ubiquitous digital communication and data storage, ensuring the security and privacy of sensitive information is paramount. Encryption systems provide a critical layer of defense against unauthorized access, interception, and tampering. This report provides an overview of encryption systems, their importance, and the current state of encryption technologies.

Fundamentals of Encryption Systems: Encryption is the process of converting plaintext into ciphertext using mathematical algorithms and keys. The encrypted data can only be deciphered back into plaintext by authorized parties possessing the corresponding decryption key. The strength of an encryption system lies in its algorithm, key length, and management practices.

2.2 Importance of Encryption Systems

Encryption systems serve several essential purposes:

- **Confidentiality**: By converting plaintext data into ciphertext, encryption ensures that only authorized parties with the appropriate decryption key can access the original information.
- **Integrity**: Encryption helps maintain the integrity of data by detecting any unauthorized modifications or tampering attempts. Even minor alterations to encrypted data can render it unreadable without the correct decryption key.
- **Authentication**: Encryption systems often incorporate digital signatures or authentication mechanisms to verify the identity of parties involved in communication or data exchange.
- **Compliance**: Many industries and regulatory bodies mandate the use of encryption to protect sensitive information and comply with data protection regulations (e.g., GDPR, HIPAA).

2.3 Types of Encryption Systems

Encryption systems can be broadly categorized into symmetric and asymmetric encryption:

2.3.1 Symmetric Encryption:

Symmetric encryption uses a single key for both encryption and decryption processes. Examples include Advanced Encryption Standard (AES), Data Encryption Standard (DES), and Triple DES (3DES).

Symmetric encryption is fast and efficient but requires secure key distribution mechanisms.

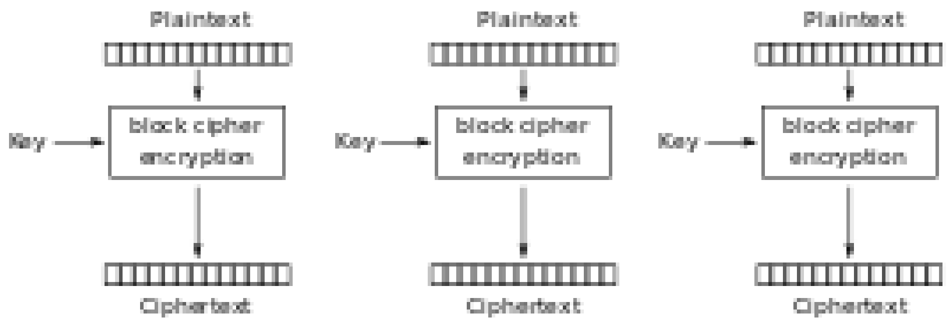
Block ciphers and stream ciphers are two fundamental types of symmetric encryption algorithms used to secure data. Let's delve into each type:

- **Block Cipher:**

Block ciphers encrypt plaintext in fixed-size blocks, typically 64 or 128 bits in length. The plaintext is divided into blocks, and each block is encrypted separately. A key is applied to each block to perform the encryption. Popular block ciphers include AES (Advanced Encryption Standard), DES (Data Encryption Standard), and 3DES (Triple DES).

Operation Modes for Block Ciphers:

1. **Electronic Codebook (ECB)**:
    - Each block of plaintext is encrypted independently using the same key. Identical plaintext blocks will result in identical ciphertext blocks, making it vulnerable to pattern analysis. See figure (2.1)



figure(2.1) Electronic Codebook (ECB) mode encryption

2. **Cipher Block Chaining (CBC)**:
    - Each plaintext block is XORed with the previous ciphertext block before encryption. This introduces diffusion, making it more resistant to pattern analysis. See figure (2.2)

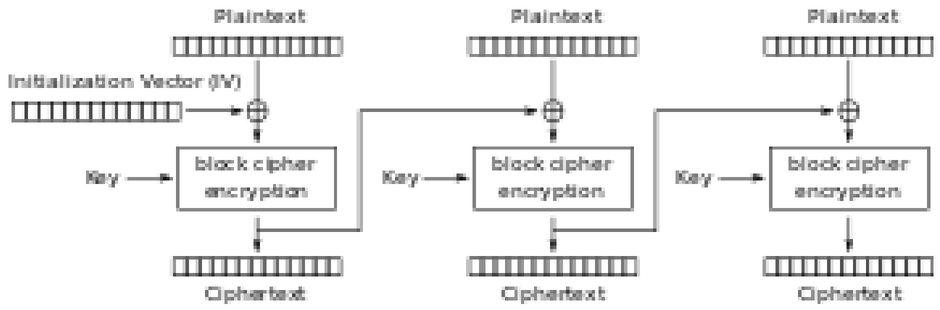


figure (2.2) Cipher Block Chaining (CBC) mode encryption

3. **Counter (CTR)**:
    - The plaintext is encrypted by XORing it with the output of a counter function. Each counter value is encrypted independently, allowing for parallel encryption and decryption.

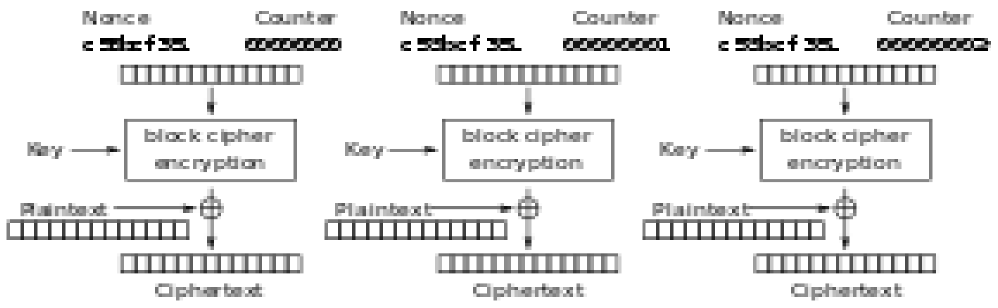


figure (2.3) Counter (CTR) mode encryption

- **Stream Cipher:**

Stream ciphers encrypt plaintext one bit or byte at a time, continuously streaming the ciphertext. They typically use a pseudorandom keystream generator combined with bitwise XOR operation to encrypt the plaintext. The keystream is often generated based on a key and an initialization vector (IV). Popular stream ciphers include RC4 and Salsa20.

Operation of Stream Cipher:

1. **Initialization**: Initialize the keystream generator with the key and an initialization vector (IV).
2. **Keystream Generation**: Generate a pseudorandom keystream based on the key and IV.
3. **Encryption**: XOR each bit or byte of the plaintext with the corresponding bit or byte of the keystream to produce ciphertext.
4. **Decryption**: XOR each bit or byte of the ciphertext with the corresponding bit or byte of the keystream to retrieve plaintext.

Stream ciphers are generally faster and require less computational resources compared to block ciphers. They are often used in applications requiring real-time encryption, such as voice and video communication.

Both block ciphers and stream ciphers play critical roles in modern cryptography, offering different advantages and use cases for securing data. The choice between them depends on factors such as security requirements, performance considerations, and application-specific needs.

2.3.1 Asymmetric Encryption:

Asymmetric encryption employs a pair of keys: a public key for encryption and a private key for decryption. Only the recipient possessing the corresponding private key can decrypt messages encrypted with the public key. Popular asymmetric encryption algorithms include RSA, Diffie-Hellman, and Elliptic Curve Cryptography (ECC).

2.4 Advancements in Encryption Technologies

Encryption technologies continue to evolve to address emerging threats and enhance security:

- **Post-Quantum Cryptography**: With the potential advent of quantum computers, which could compromise traditional encryption algorithms, research is underway to develop quantum-resistant encryption methods known as post-quantum cryptography.
- **Homomorphic Encryption**: Homomorphic encryption allows computations to be performed on encrypted data without the need for decryption, offering privacy-preserving solutions for cloud computing and data analytics.
- **End-to-End Encryption (E2EE)**: E2EE ensures that data is encrypted at the source and can only be decrypted by the intended recipient, eliminating the risk of interception or surveillance during transmission.

2.5 Applications of Encryption Systems:

- **Data Protection:** Encryption safeguards sensitive data at rest (stored on devices or servers) and in transit (during communication over networks).
- **Secure Communication:** Encryption ensures the confidentiality of emails, instant messages, voice calls, and video conferences.
- **E-commerce Security:** Encryption secures online transactions, protecting financial details and personal information.
- **Cloud Security:** Encryption safeguards data stored in cloud environments, preventing unauthorized access.

2.6 Challenges and Considerations

Despite the benefits of encryption, several challenges persist:

- **Key Management**: Securely managing encryption keys, including generation, distribution, storage, and revocation, remains a complex task, especially in large-scale deployments.
- **Performance Overhead**: Strong encryption algorithms can introduce computational overhead, impacting system performance and response times, particularly in resource-constrained environments.
- **Regulatory Compliance**: Navigating regulatory requirements and compliance standards while implementing encryption solutions can pose legal and operational challenges for organizations.

2.7 type of attack

Positive and negative attacks can refer to various concepts across different domains, including cybersecurity, social engineering, psychology, and more. Below, I'll provide explanations and examples of positive and negative attacks in the context of cybersecurity along with relevant sources for further reading:

2.7.1 Positive Attacks:

1. **White-Hat Hacking**:
   - Description: Ethical hacking performed by security professionals to identify vulnerabilities in systems, networks, or applications.
   - Example: Penetration testing, vulnerability assessment, and security audits.
   - Source: "What is White Hat Hacking? Definition, Types, and Tools" on Varonis.
2. **Security Patching**:
   - Description: Proactively applying updates or patches to fix known vulnerabilities in software, hardware, or systems.
   - Example: Regularly updating operating systems, applications, and firmware.
   - Source: "The Importance of Patch Management and How to Achieve It" on Security Magazine.
3. **Security Awareness Training**:
   - Description: Educating users and employees aBoat cybersecurity best practices, threats, and how to recognize and respond to them.
   - Example: Phishing awareness training, password hygiene training.

- Source: "The Importance of Security Awareness Training for Your Staff" on Comodo.

2.7.2 Negative Attacks:

1. **Malware**:
    - Description: Malicious software designed to disrupt, damage, or gain unauthorized access to computer systems or data.
    - Example: Viruses, worms, trojans, ransomware.
    - Source: "What is Malware? Definition, Examples, and Prevention" on Norton.
2. **Phishing**:
    - Description: Social engineering attack where attackers impersonate legitimate entities to trick individuals into revealing sensitive information or performing actions.
    - Example: Fake emails, websites, or messages claiming to be from a bank, government agency, or company.
    - Source: "Phishing: What Is It and How to Protect Yourself From It" on Kaspersky.
3. **Distributed Denial of Service (DDoS)**:
    - Description: Attack where multiple compromised systems flood a targeted system or network with excessive traffic, rendering it inaccessible.
    - Example: Botnets launching coordinated DDoS attacks against websites or online services.
    - Source: "Distributed Denial of Service (DDoS) Attacks" on Imperva.
4. **Insider Threats**:
    - Description: Attacks or breaches caused by individuals within an organization who misuse their access privileges or intentionally harm the organization's security.
    - Example: Employees stealing data, leaking sensitive information, or installing malware.
    - Source: "Insider Threats: Types, Challenges, Prevention, and More" on Trend Micro.

These sources provide in-depth information on positive and negative attacks in the context of cybersecurity, helping you understand various attack vectors and preventive measures.

2.8 Lightweight cryptography

Lightweight cryptography refers to cryptographic algorithms and protocols designed specifically for resource-constrained environments, such as embedded systems, IoT devices, RFID tags, and smart cards. These algorithms prioritize efficiency, low power consumption, and small code size while still providing adequate security. Below is a report on lightweight cryptography, including its importance, characteristics, and examples of lightweight algorithms, along with relevant sources for further reading.

2.8.1 Title: Lightweight Cryptography: Balancing Efficiency and Security

- **Introduction:** Lightweight cryptography has emerged as a critical area of research due to the proliferation of resource-constrained devices and the increasing demand for secure communication and data protection in such environments. These devices, including IoT sensors, wearable devices, and smart cards, often operate under strict constraints of power, processing capabilities, and memory. Traditional cryptographic algorithms may be too computationally intensive or require excessive memory, making them unsuitable for these devices. Lightweight cryptography addresses these challenges by providing efficient and secure cryptographic solutions tailored to the constraints of these devices.

- **Importance of Lightweight Cryptography:** The importance of lightweight cryptography stems from its ability to enable secure communication and data protection in resource-constrained environments. Key reasons for the significance of lightweight cryptography include:

    1. **Resource Efficiency**: Lightweight algorithms are designed to operate efficiently in terms of power consumption, processing requirements, and memory usage, making them suitable for devices with limited resources.
    2. **Security**: Despite their lightweight nature, these algorithms aim to provide adequate levels of security, ensuring confidentiality, integrity, and authenticity of data.
    3. **Versatility**: Lightweight cryptography offers versatility, catering to a wide range of applications such as IoT, RFID, smart cards, and embedded systems.
    4. **Standardization Efforts**: Standardization bodies and research communities are actively working on developing and standardizing lightweight cryptographic algorithms, further highlighting their importance.

- ❖ **Characteristics of Lightweight Cryptography:** Lightweight cryptographic algorithms exhibit several key characteristics:

  1. **Efficiency**: Lightweight algorithms prioritize efficiency in terms of computation, power consumption, and memory usage.
  2. **Low Complexity**: These algorithms feature low computational and implementation complexity to minimize resource requirements.
  3. **Compactness**: Lightweight algorithms have small code and data size, making them suitable for deployment on resource-constrained devices.
  4. **Security**: Despite being lightweight, these algorithms provide sufficient security against various cryptographic attacks, ensuring the confidentiality and integrity of sensitive data.
  5. **Adaptability**: Lightweight algorithms are adaptable to different application scenarios and can be tailored to specific requirements.

2.8.2 Examples of Lightweight Cryptographic Algorithms:

Several lightweight cryptographic algorithms have been proposed and developed by researchers and standardization bodies. Some notable examples include:

1. Present Algorithm

- **Introduction**: Present is a lightweight block cipher designed by A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. B. Robshaw, Y. Seurin, and C. Vikkelsoe. It was introduced in 2007 and has gained attention for its simplicity and efficiency.
- **Features**:
  - **Block Size**: 64 bits.
  - **Key Size**: 80 or 128 bits.
  - **Round Function**: Based on a substitution-permutation network (SPN) structure, consisting of key addition, a nonlinear layer, a linear layer, and a key schedule.
  - **Number of Round:**
    - For a 80-bit key: 31 rounds
    - For a 128-bit key: 32 rounds
    - For a 192-bit key: 33 rounds
- **Security**: Present has undergone extensive analysis and is considered secure against various cryptanalytic attacks. It provides a good balance between security and performance, making it suitable for constrained environments. See to figure (2.4)

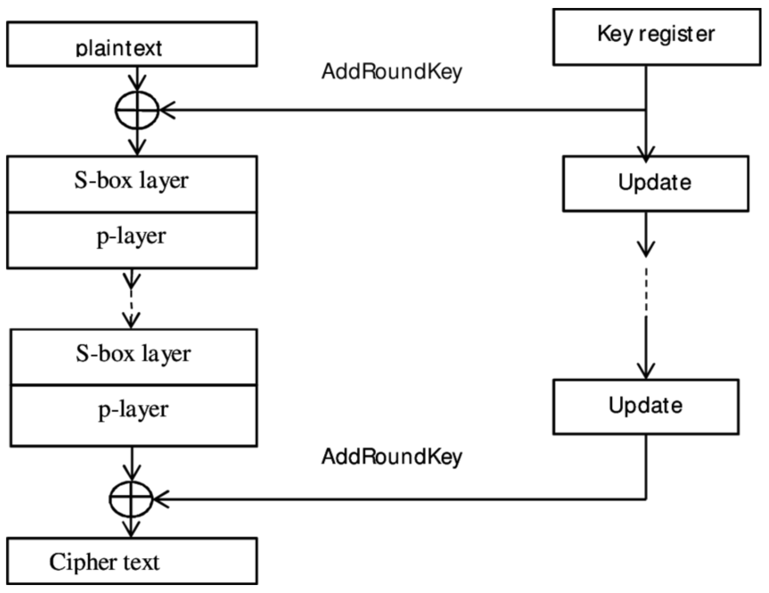


Figure (2.4): Block-diagram-of-present-cipher

2. Simon Algorithm

- **Introduction**: Simon is a family of lightweight block ciphers designed by the NSA. It emphasizes simplicity, efficiency, and security, making it suitable for resource-constrained environments.
- **Features**:
  - **Block Size**: Variable (e.g., 32, 48, 64, 96, 128 bits).
  - **Key Size**: Variable (e.g., 64, 72, 96, 128, 144, 192, 256 bits).
  - **Round Function**: Utilizes simple bitwise operations, such as XOR, AND, and cyclic shifts.
  - **Number of Round:** The number of rounds depends on the specific configuration. Here are some common configurations:
    - Simon32/64: 32 rounds
    - Simon48/72: 36 rounds
    - Simon48/96: 36 rounds
    - Simon64/96: 42 rounds
    - Simon64/128: 44 rounds
- **Security**: Simon has undergone rigorous cryptanalysis, demonstrating its resistance against various attacks. It offers a high level of security while maintaining efficiency. See to figure (2.5)

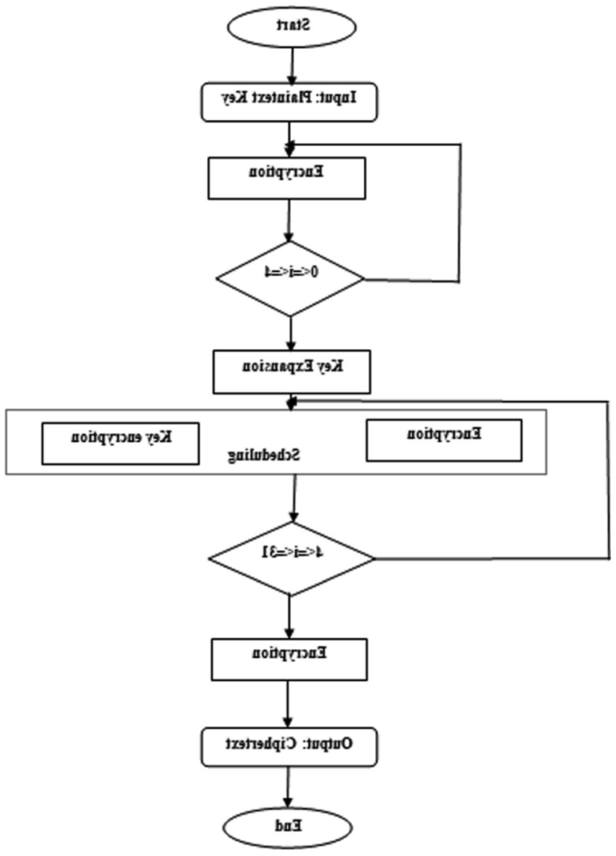


Figure (2.5): Block-diagram-of-Simon-cipher

3. Speck Algorithm

- **Introduction**: Speck is another family of lightweight block ciphers developed by the NSA. It aims to provide high performance and security in both hardware and software implementations.
- **Features**:
  - **Block Size**: Variable (e.g., 32, 48, 64 bits).
  - **Key Size**: Variable, similar to Simon.
  - **Round Function**: Employs a combination of bitwise rotations, XOR operations, and modular addition.
  - **Number of Round:** The number of rounds depends on the block size and key size. Common configurations include:
    - Speck32/64: 22 rounds
    - Speck48/72: 22 rounds
    - Speck48/96: 23 rounds
    - Speck64/96: 26 rounds

- **Security**: Speck has been subjected to thorough analysis, demonstrating its resilience against various cryptanalytic techniques. It offers a good balance between security and efficiency. to figure (2.6)

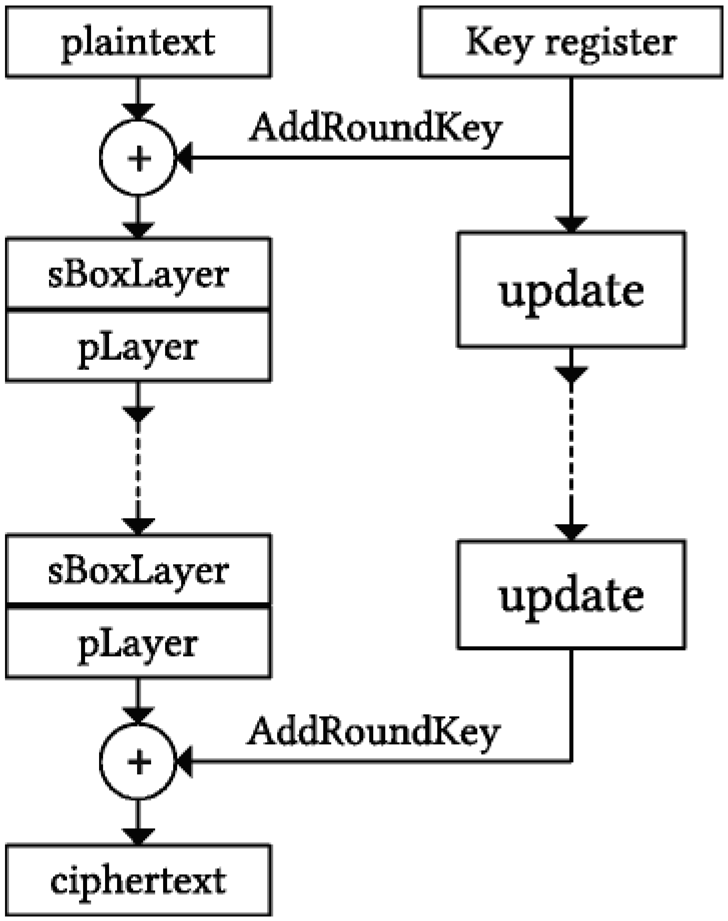


Figure (2.6): Block-diagram-of-Simon-cipher

Chapter Three
The speak algorithm and experimental results

3.1 Speck Cryptography

SPECK is a lightweight block cipher designed by the National Security Agency (NSA) for use in securing electronic communication and data privacy. It is designed to provide efficient and secure encryption of data while maintaining simplicity and speed in its implementation. SPECK operates on blocks of 64 or 128 bits in size, and uses a combination of simple operations such as bitwise XOR, addition modulo 2^32, and bitwise shifts to achieve its encryption and decryption functions. Due to its lightweight nature and strong encryption properties, SPECK has gained popularity for use in resource-constrained devices and applications where efficiency and security are paramount. See figure(3.1)

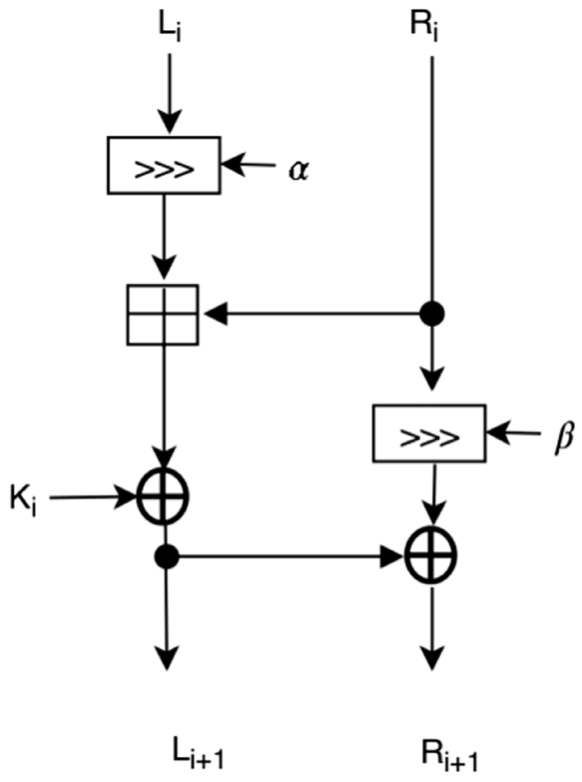


Figure (3.1): speck round function

* Now the following is the encryption function

  Algorithm Speck Encrypt (ctx, plaintext, ciphertext):

Input:

ctx: Speck context containing encryption parameters and key schedule.

plaintext: Array of two 64-bit integers representing the plaintext.

ciphertext: Array of two 64-bit integers representing the ciphertext.

Output:

None. The ciphertext is stored in the provided ciphertext array.

Procedure:

1. Copy the first 64-bit integer of the plaintext to the first 64-bit integer of the ciphertext.

2. Copy the second 64-bit integer of the plaintext to the second 64-bit integer of the ciphertext.

3. Iterate over the number of rounds specified in the context:

   a. Call the speck_round function with arguments:

      - The address of the second 64-bit integer of the ciphertext.

      - The address of the first 64-bit integer of the ciphertext.

      - The address of the i-th key schedule entry in the context.

∗ Now, the algorithm for the round function

```
Algorithm speck_round(x, y, k):
    Input:
        x: Pointer to a 64-bit integer representing one half of the block.
        y: Pointer to a 64-bit integer representing the other half of the block.
        k: Pointer to a 64-bit integer representing the round key.

    Output:
        None. The updated values of x and y are stored at the provided memory addresses.

    Procedure:
        1. Right rotate the value pointed to by x by 8 bits.
        2. Add the value pointed to by y to the value pointed to by x.
        3. XOR the value pointed to by x with the value pointed to by k.
        4. Left rotate the value pointed to by y by 3 bits.
        5. XOR the value pointed to by y with the updated value pointed to by x.
```

Etailed description of the Speck algorithm:

3.1.1 Key Features:

1. **Block Size**: Speck supports two block sizes: 64 bits and 128 bits.
2. **Key Size**: The key size can be either 64, 96, or 128 bits.
3. **Round Structure**: Speck employs a Feistel network structure with a variable number of rounds depending on the block and key sizes.
4. **Key Schedule**: Speck uses a key schedule algorithm based on a simple round function.

3.1.2 Algorithm Overview:

3.1.2.1 Initialization:

- Input: Plaintext block (64 bits), Key (64, 96, or 128 bits).
- Output: Encrypted block.

3.1.2.2 Key Expansion:

- The key is expanded into an array of round keys using a simple key schedule algorithm.

3.1.2.3 Encryption:

- Speck operates on the plaintext block using the round keys generated from the key expansion.
- The encryption process consists of a series of rounds, typically between 22 to 34 rounds depending on the block and key sizes.
- Each round involves a combination of bitwise XOR, left and right circular shifts, and modular addition operations.
- The round function mixes the plaintext with the current round key using bitwise XOR operations, rotates the resulting value, and adds the round key again.
- The number of rounds is chosen to achieve the desired level of security while maintaining efficiency.

3.1.2.4 Decryption:

- Speck decryption is essentially the reverse process of encryption, with the round keys applied in reverse order.

3.1.3 Security:

Speck has undergone extensive analysis by cryptographers to ensure its security. It has been scrutinized for resistance against various attacks, including differential and linear cryptanalysis. While it may not have the same level of scrutiny or assurance as more widely adopted ciphers like AES, it is considered secure for many practical applications, especially those requiring lightweight and efficient cryptography.

3.1.4 Implementation:

Speck's simplicity makes it suitable for implementation in hardware or software on resource-constrained devices. It requires minimal memory and computational resources compared to more complex ciphers like AES. However, efficient implementations must take care to protect against side-channel attacks, particularly on hardware platforms.

3.2 Performance Results

In this section, we present the results of execution the speck algorithm that encrypt / decrypt different gray scale image. Both, execution time and throughput are measured for each image size as will as histogram for both original image and encrypt image see table (1)

Table (3.1): Performance Results

Image name	Image size	Encryption time seconds	Throughput Gbits/s
Cameraman	320 * 320	0.006828	0.307140
Boat	512*512	0.007207	0.290988
Fruit	305 * 305	0.007231	0.290022

3.3 Histogram analysis results

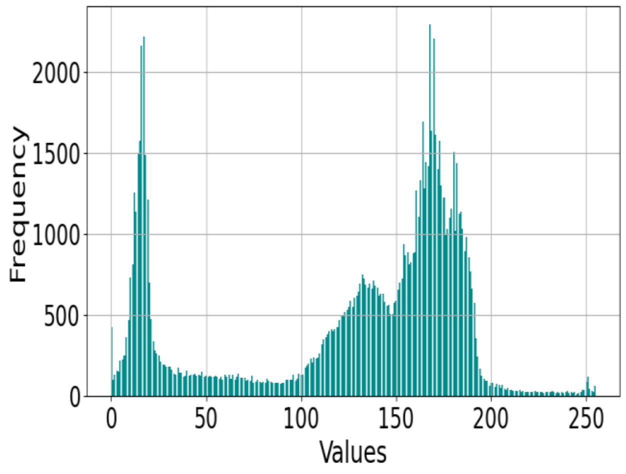
The histogram analysis must be close to the uniform distribution. The best threshold of uniform distribution that the good cipher method must be is very close to image size /256.

For example, in the case of gray scale image that have size of 512*512 the histogram must equal to 1024.

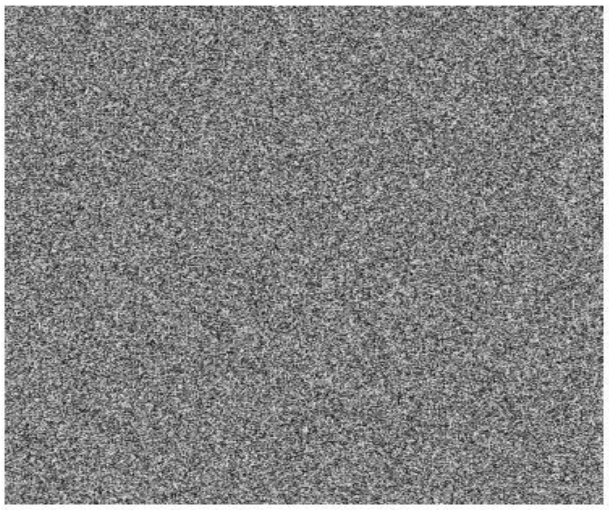
1. The histogram of cameraman image

The histogram for the plain and the encrypted images is computed and draws as follows:

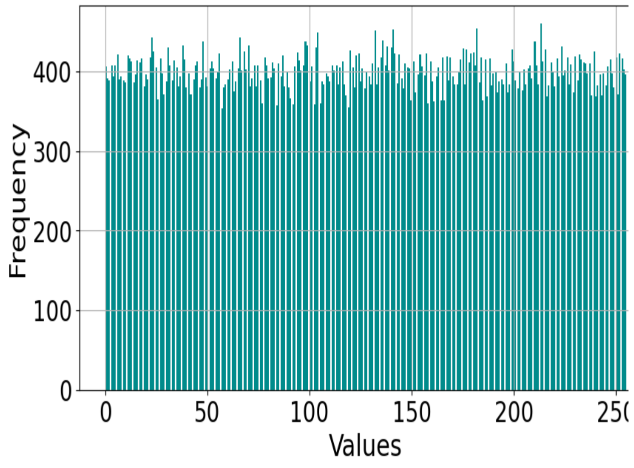
A. Plain Image cameraman



B. histogram Plain Image cameraman



C. Encrypted cameraman Image



D. histogram of Encrypted cameraman Image

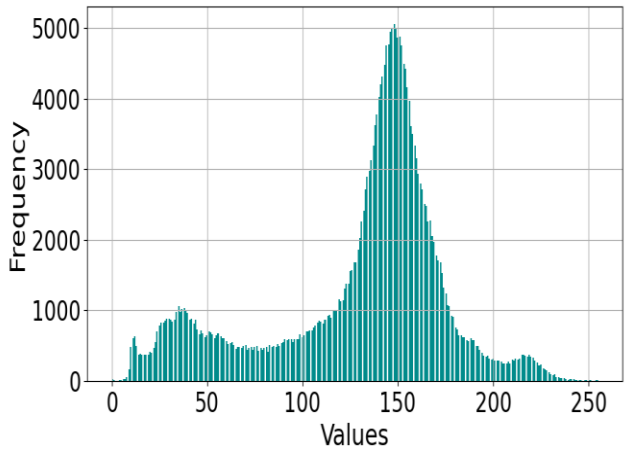
Figure (3.2): The histogram analysis for the cameraman image

The histogram analysis results of cameraman cipher image of size 320*320 is close to 400, which is mean that is close to the desired histogram distribution.

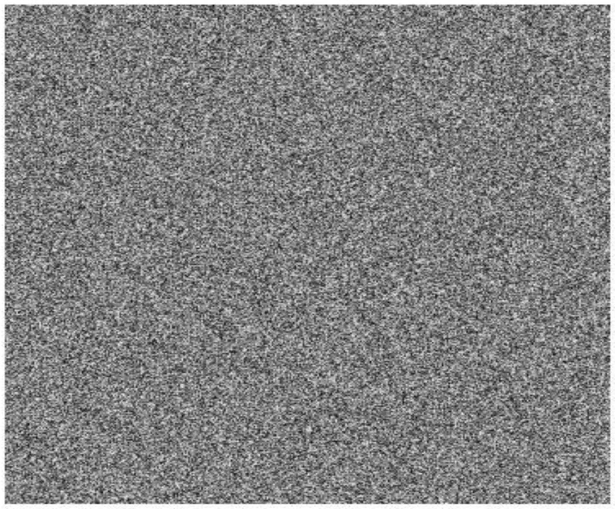
2. The histogram of boat image

The histogram for the plain and the encrypted images is computed and draws as follows:

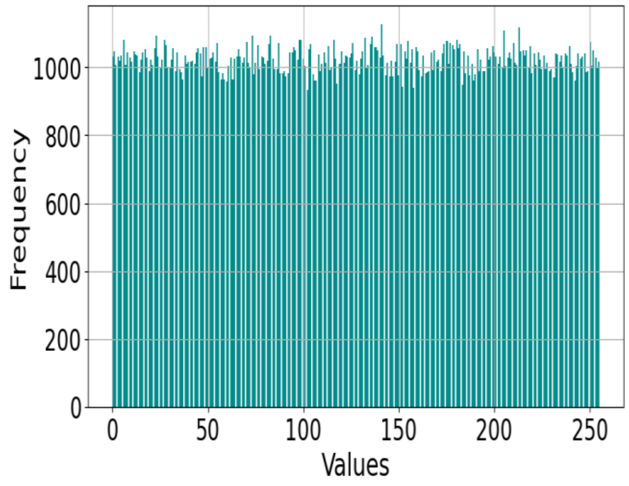
A. Plain Image Boat



B. histogram Plain Image Boat



C. Encrypted Boat Image



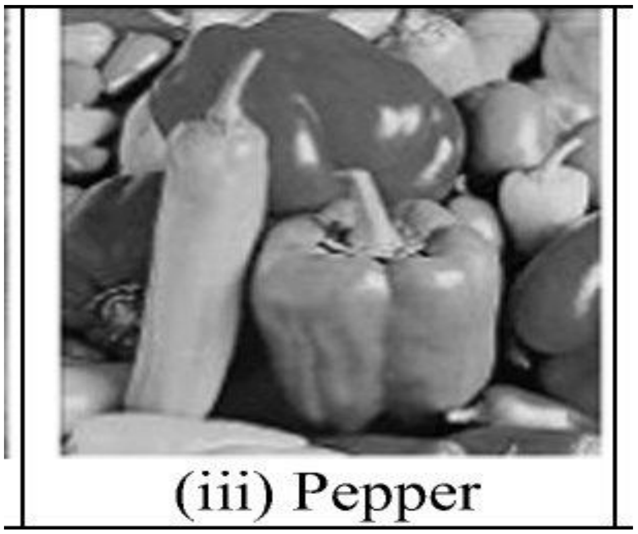
D. histogram of Encrypted Boat Image

Figure (3.2): The histogram analysis for the Boat image

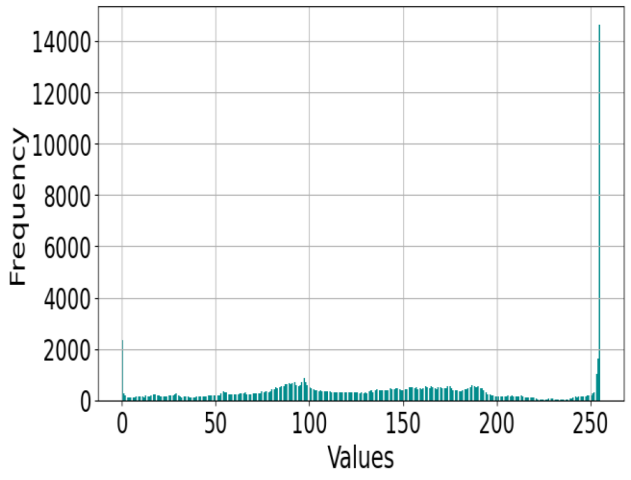
The histogram analysis results of Boat cipher image of size 512*512 is close to 1024, which is mean that is close to the desired histogram distribution.

The histogram of fruit image

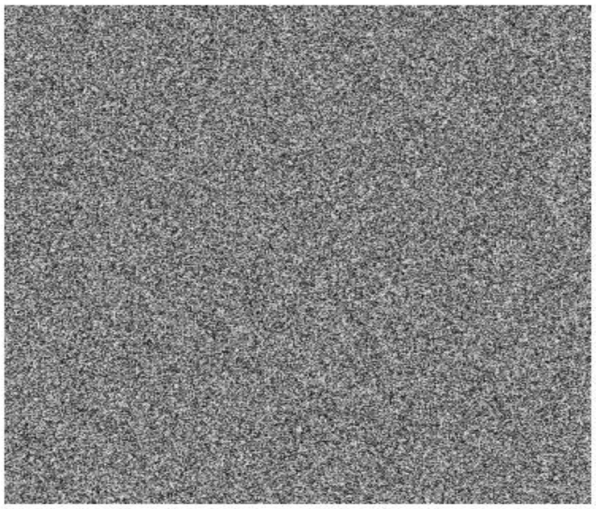
The histogram for the plain and the encrypted images is computed and draws as follows:



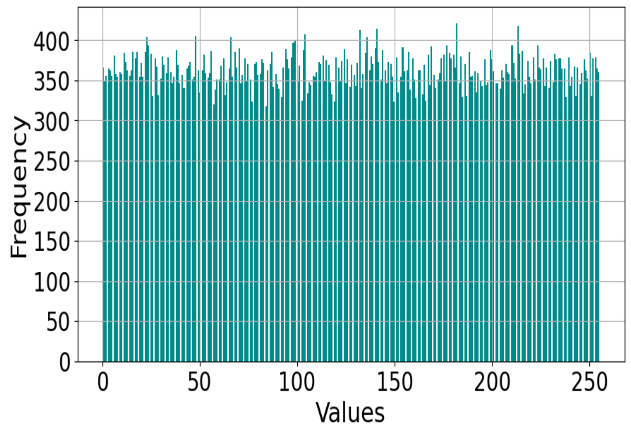
A. Plain Image fruit



B. histogram Plain Image fruit



C. Encrypted fruit Image



D. histogram of Encrypted fruit Image

Figure (3.2): The histogram analysis for the fruit image

The histogram analysis results of fruit cipher image of size 305*305 is close to 363.38, which is mean that is close to the desired histogram distribution.

Chapter four
Conclusions and future works

4.1 Conclusions:

1. The project successfully implemented the encryption of grayscale images using the Speck algorithm.

2. Efficiency was assessed by calculating execution time and throughput.

3. Productivity was evaluated based on the encryption process and comparison with original images.

4. The histogram results obtained indicate the required security level for the Speck algorithm when encrypting the plain images.

4.2 Future works

1. Explore the possibility of optimizing the encryption algorithm for faster execution.

2. Conduct further analysis on the security level of the encryption method used.

3. Experiment with encrypting images of different types (e.g color images) and sizes to test the scalability and performance of the algorithm.

4. Consider implementing additional features, such as decryption validation and error handling.

Sources:

1. Stallings, W. (2019). *Cryptography and Network Security: Principles and Practice* (8th ed.). Pearson Education.
2. Schneier, B. (2015). *Applied Cryptography: Protocols, Algorithms, and Source Code in C* (2nd ed.). John Wiley & Sons.
3. National Institute of Standards and Technology (NIST). (2020). *Recommendations for Key Management: Part 1: General*. NIST Special Publication 800-57.
4. Diffie, W., & Hellman, M. E. (1976). *New Directions in Cryptography*. IEEE Transactions on Information Theory, 22(6), 644-654.
5. Dijk, M., Gentry, C., Halevi, S., & Vaikuntanathan, V. (2010). *Fully Homomorphic Encryption over the Integers*. Advances in Cryptology – EUROCRYPT 2010.
6. National Institute of Standards and Technology (NIST): NIST Encryption Standards
7. International Organization for Standardization (ISO): ISO/IEC 18033: Information Technology - Security Techniques - Encryption Algorithms
8. Electronic Frontier Foundation (EFF): Surveillance Self-Defense - Encryption
9. Schneier on Security: Cryptography
10. A. Bogdanov et al., "PRESENT: An Ultra-Lightweight Block Cipher," in Cryptographic Hardware and Embedded Systems - CHES 2007, 2007. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-540-74735-2_31
11. M. Abdelraheem and A. Fathy, "LEDA: Lightweight Encryption and Data Authentication for Internet of Things Applications," in 2018 21st Saudi Computer Society National Computer Conference (NCC)., 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8584962
12. A. Bogdanov et al., "Spongent: A Lightweight Hash Function," in Selected Areas in Cryptography - SAC 2011, 2011. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-642-23951-9_13
13. "The Block Cipher PRESENT" by A. Bogdanov et al.
14. "The Simon and Speck Families of Lightweight Block Ciphers" by Ray Beaulieu et al.
15. Official specifications and documentation for Present, Simon, and Speck algorithms.
16. Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., & Weeks, B. (2015). "The SIMON and SPECK Families of Lightweight Block Ciphers." Cryptology ePrint Archive, Report 2015/585.

17. Banik, S., & Isobe, T. (2016). "Differential and Linear Cryptanalysis of Reduced Round SIMON and SPECK." In International Conference on Cryptology in Africa (pp. 3-22). Springer, Cham.

18. National Institute of Standards and Technology (NIST). NIST Cryptographic Toolkit.

19. Schneier, B. (2013). "Speck: A New NSA Encryption Algorithm." Schneier on Security

20. Dinur, I., Golić, J., & Shamir, A. (2018). "Cube Attacks on Cryptographic Implementations." Cryptology ePrint Archive, Report 2018/827