IACR Transactions
ISSN XXXX-XXXX, Vol. 0, No. 0, pp. 1–22. DOI:XXXXXXXX

An Analytical Study of the SQUARE Block Cipher: Structure, Properties, and Cryptanalysis

Nikhil Kumar Shrey[1], Aayush Kataria[2] and Sudeep Ranjan Sahoo[3]

Indian Institute of Technology Bhilai, Bhilai, India
nikhilkumar@iitbhilai.ac.in,
aayushka@iitbhilai.ac.in,
sudeepranjan@iitbhilai.ac.in

Abstract. This paper analyzes the SQUARE block cipher, a 128-bit symmetric key cipher that inspired the design of AES. The study delves into the structure and key scheduling of SQUARE, its S-box properties, and cryptographic parameters, including the linear approximation table (LAT) and differential distribution table (DDT). We perform differential and integral cryptanalysis on reduced-round versions, identify potential vulnerabilities, and discuss the cipher's implementation. A detailed comparison with AES highlights the similarities and distinctions between the two ciphers.

Keywords: SQUARE Block Cipher · Cryptanalysis · S-box Analysis · LAT · DDT

1 Introduction

1.1 Background and Historical Context

The SQUARE block cipher was proposed in 1997 by Joan Daemen, Lars Knudsen, and Vincent Rijmen as a candidate for a high-security encryption algorithm. Designed to operate on a 128-bit block and 128-bit key, SQUARE utilizes a structure that would later influence the design of the Advanced Encryption Standard (AES). It is a symmetric key cipher that introduced key innovations in cryptographic design, such as the use of the *wide trail design strategy*, which was explicitly designed to resist differential and linear cryptanalysis.

SQUARE's design is notable for its simplicity and efficiency, with an emphasis on the secure diffusion of information through the cipher's transformations. The cipher comprises eight rounds, each incorporating a set of four operations: substitution, permutation, key addition, and linear transformation. Despite its early promise, the adoption of SQUARE was overshadowed by AES, which eventually became the standard block cipher for secure data encryption, after SQUARE's internal weaknesses were identified in subsequent cryptanalysis studies.

The SQUARE cipher, while not widely used in practice, is an important cryptographic algorithm because of its role in shaping the development of AES and its contribution to the study of block cipher design.

1.2 Significance of SQUARE in Cryptography

SQUARE holds an important place in cryptography, primarily for its conceptual and design contributions to modern symmetric key encryption algorithms. Its development marked a significant step in the evolution of block ciphers, particularly in addressing concerns related

to differential and linear cryptanalysis. The cipher introduced the *Wide Trail Design Strategy*, which aims to ensure that the probability of successful attacks remains low even when the number of rounds is reduced. This strategy focuses on spreading the influence of each bit of the plaintext throughout the entire ciphertext by using both nonlinear and linear operations.

The Wide Trail Design Strategy is particularly influential in the design of AES, where similar principles were applied to ensure security against attacks like differential and linear cryptanalysis. Although SQUARE itself was not adopted as the standard, its design principles have been widely influential in the development of secure ciphers.

SQUARE also serves as a pedagogical tool for understanding the trade-offs between design simplicity and cryptographic strength. Its legacy includes the lessons learned from its weaknesses in early cryptanalysis, which led to the refinement of AES and other contemporary encryption algorithms.

1.3 Objectives of the Paper

The objectives of this paper are as follows:

- To analyze the structure and design of the SQUARE block cipher, focusing on its round transformations, key scheduling, and S-box design.
- To perform a detailed cryptanalysis of SQUARE, including differential, linear, and integral attacks, and identify potential vulnerabilities in its design.
- To compare SQUARE with AES, highlighting the similarities and differences in design, cryptographic strength, and security.
- To explore the significance of SQUARE in the evolution of modern cryptography, particularly its influence on AES.

This paper will provide both theoretical and practical insights into the SQUARE cipher, contributing to the understanding of block cipher design principles and cryptographic analysis.

—

2 Structure of SQUARE

The SQUARE block cipher is an iterated block cipher with a block size and key size of 128 bits each. The structure of the cipher consists of a series of transformations applied to the state matrix in multiple rounds. These transformations ensure high diffusion and security, making the cipher resistant to various cryptanalytic attacks, such as differential and linear cryptanalysis. In this section, we explain the components that make up the structure of SQUARE, starting from the state matrix to the round transformations and the overall encryption process.

2.1 Overview of the Cipher Design

SQUARE operates on a 128-bit block and key size, with the state represented as a 4×4 matrix of bytes. This matrix is used to hold the intermediate state during encryption, where each element of the matrix is a byte. The state matrix is initialized with the plaintext, and the key schedule generates round keys that are used in each round of the encryption process.

$$\text{State} = \begin{bmatrix} a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} \\ a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,0} & a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,0} & a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix}$$

Each element $a_{i,j}$ in the state matrix represents a byte of the 128-bit block. The cipher uses 8 rounds of transformations to modify this state matrix, with each round applying a set of operations to achieve diffusion and confusion.

2.2 Round Transformations

Each round of the SQUARE cipher consists of four transformations applied in sequence to the state matrix. These transformations are designed to provide both diffusion and nonlinearity, ensuring resistance against cryptanalytic attacks. The four transformations are:

- **Linear Transformation (θ)**: A matrix multiplication that operates on each row of the state matrix.
- **Nonlinear Substitution (γ)**: Byte-wise substitution using an S-box.
- **Byte Permutation (π)**: Transposition of the state matrix by swapping rows and columns.
- **Round Key Addition (σ)**: XORing the state with the round key.

The round transformation for each round t can be represented as:

$$\text{State}_{t+1} = \sigma(k_t) \circ \pi \circ \gamma \circ \theta(\text{State}_t)$$

Where: - $\sigma(k_t)$ represents the round key addition, - π is the byte permutation, - γ is the nonlinear substitution, - θ is the linear transformation.

2.2.1 Linear Transformation (θ)

The linear transformation θ operates separately on each row of the state matrix and ensures high diffusion of the bits across the state. Transformation is defined as a polynomial multiplication over $GF(2^8)$:

$$b_{i,j} = c_0 \cdot a_{i,0} \oplus c_1 \cdot a_{i,1} \oplus c_2 \cdot a_{i,2} \oplus c_3 \cdot a_{i,3}$$

Where c_0, c_1, c_2, c_3 are constants derived from the polynomial $c(x) = 2x \oplus x^2 \oplus 3x^3$. This transformation ensures that the influence of each byte is spread across the entire row, promoting diffusion.

2.2.2 Nonlinear Substitution (γ)

The nonlinear substitution step involves applying an 8-bit S-box to each byte in the state matrix. The substitution is designed to ensure nonlinearity and provide resistance to differential cryptanalysis. For a byte $a_{i,j}$, the substitution is defined as:

$$b_{i,j} = S(a_{i,j})$$

Where $S(x)$ is an invertible 8-bit substitution table (S-box) used for the nonlinear substitution.

2.2.3 Byte Permutation (π)

The byte permutation π is a simple transposition that swaps the rows and columns of the state matrix. This step is designed to increase the diffusion of the cipher. The permutation is defined as:

$$b_{i,j} = a_{j,i}$$

This transformation is an involution, meaning that applying it twice results in the original state matrix. Therefore, $\pi^{-1} = \pi$.

2.2.4 Round Key Addition (σ)

The round key addition involves XORing the state matrix with the round key k_t. This operation ensures that the key influences the state at every round, providing the necessary security. The round key addition is defined as:

$$b = A \oplus k_t$$

Where A is the current state, and k_t is the round key for round t.

2.3 Round Structure

The overall structure of each round can be visualized as the application of the four transformations to the state matrix in sequence. The diagram below illustrates the sequence of transformations applied in a single round of the SQUARE cipher:

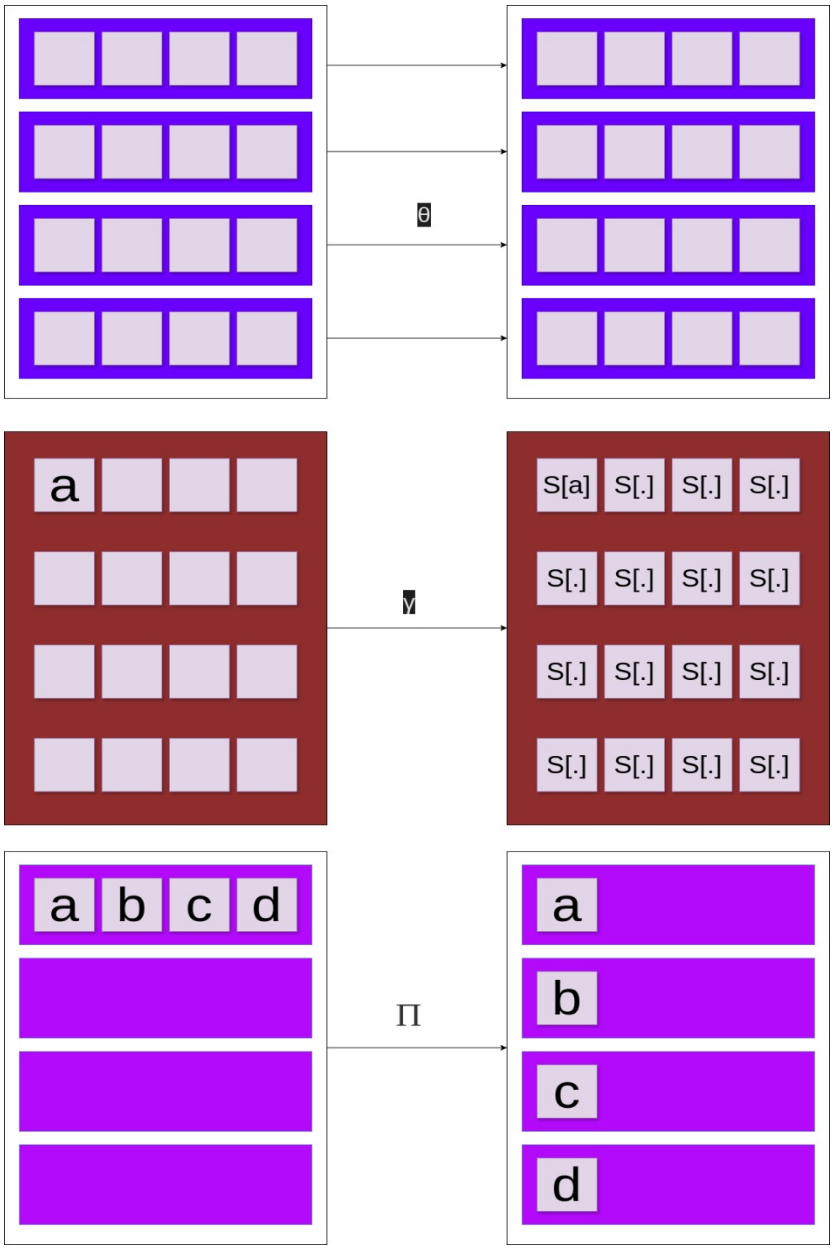

Figure 1: Round structure of SQUARE cipher.

As shown in the diagram, the state undergoes the following transformations in each round: 1. Apply linear transformation (θ), 2. Apply nonlinear substitution (γ), 3. Apply byte permutation (π), 4. Add round key (σ).

2.4 Algorithm Workflow

The complete encryption process for SQUARE involves applying the round transformation to the state matrix for 8 rounds, as well as performing an initial key addition and final inverse transformation. The overall workflow of the algorithm can be summarized in the following pseudocode:

```
Input: Plaintext P, Key K
Output: Ciphertext C

1. Initialize state S = P      K      // XOR plaintext with the
    key
2. for round r = 1 to 8:
   3. Apply Linear Transformation    (S)
   4. Apply Nonlinear Substitution    (S)
   5. Apply Byte Permutation    (S)
   6. Add Round Key: S = S      K_r    // XOR with round key
3. Output C = S  // Final ciphertext

Decryption involves reversing the transformations in reverse
   order with the inverse operations.
```

Listing 1: Pseudocode for SQUARE Encryption

In this pseudocode: - The “, “, “, and “ transformations are applied in sequence. - The final ciphertext C is the result of applying these transformations for 8 rounds.

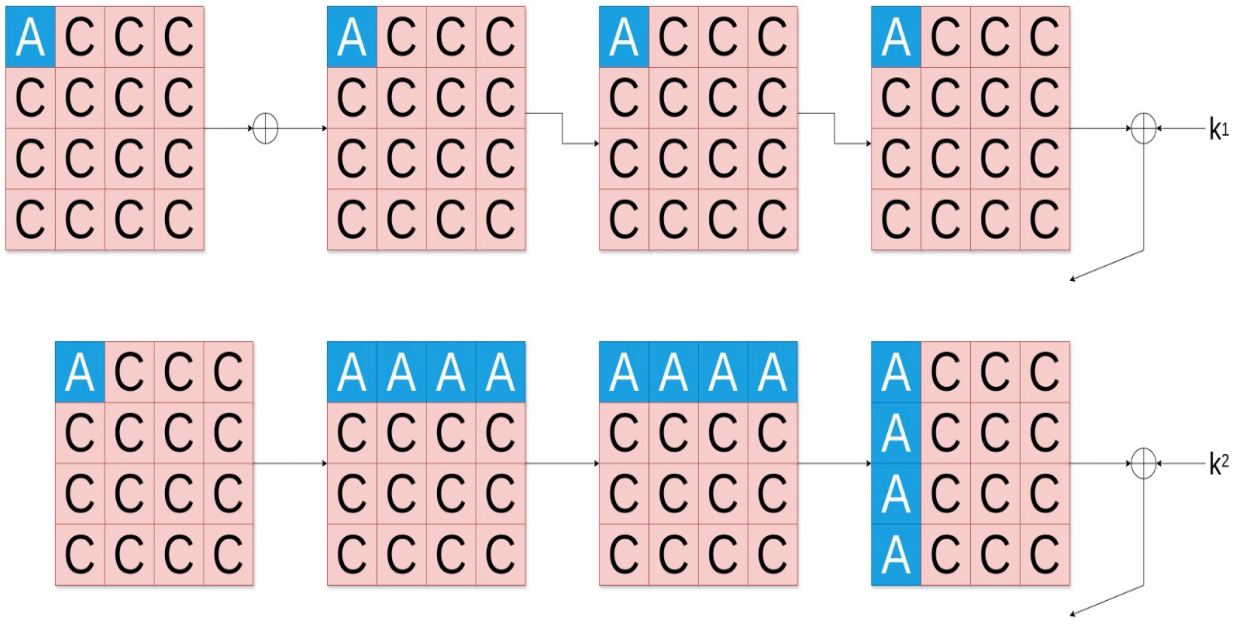

Figure 2: SQUARE cipher.

—

3 Key Scheduling

The key scheduling process in the SQUARE block cipher is responsible for generating the round keys from the initial 128-bit key, which are then used during each round of encryption. The key schedule ensures that the round keys are different from each other and provide enough diffusion to resist various cryptanalytic attacks. The process is designed to guarantee that the relationship between the round keys is complex and non-linear, thereby preventing any straightforward attack on the key.

3.1 Overview of Key Expansion

The SQUARE cipher uses a 128-bit key K, which is initially divided into four 32-bit words, denoted as w_0, w_1, w_2, w_3. These words represent the initial round key, and they are used to generate 8 distinct round keys for the 8 rounds of encryption.

$$K = (w_0, w_1, w_2, w_3)$$

The initial key is expanded to produce 8 round keys, $k_0, k_1, \ldots, k_8$, which are used during the encryption process. Each round key is derived iteratively through a combination of transformations, including rotations, XOR operations with round constants, and recursive use of previous round key values. This process ensures that the round keys are spread out and sufficiently diffused.

3.2 Key Generation Process

The round key generation in SQUARE involves several steps, including byte rotation, XOR with round constants, and recursive transformations. The general key expansion rule is as follows:

$$w_t = w_{t-1} \oplus \text{rotation}(w_{t-1}) \oplus C_t$$

Where: - $\text{rotation}(w_{t-1})$ refers to rotating the word w_{t-1} by one byte to increase diffusion. - C_t represents the round constant for round t.

The round constants C_t are chosen iteratively, and they ensure that the round keys are not symmetric, making it harder for attackers to find patterns that could lead to efficient key recovery. The round constants are defined as:

$$C_0 = 1, \quad C_t = 2 \cdot C_{t-1} \quad \text{for} \quad t > 0$$

This definition allows the constants to grow exponentially, contributing to the diffusion of the round keys.

The key generation process for the round keys $k_0, k_1, \ldots, k_8$ can be summarized as:

$$\begin{aligned}
k_0 &= K \\
k_1 &= w_0 \oplus w_1 \oplus w_2 \oplus w_3 \\
k_2 &= \text{rotation}(k_1) \oplus C_1 \\
k_3 &= \text{rotation}(k_2) \oplus C_2 \\
&\vdots \\
k_8 &= \text{rotation}(k_7) \oplus C_7
\end{aligned}$$

The recursive use of w_{t-1} in each iteration ensures that each round key is dependent on all previous keys and the initial key, which increases the complexity and security of the key scheduling.

3.3 Security Properties of Key Schedule

The key schedule plays an important role in the overall security of the cipher. Key security is enhanced by the following features:

1. **High Diffusion**: The use of rotations and XOR operations guarantees that the influence of each bit in the key spreads rapidly across the rounds, making it difficult for an attacker to determine the key from partial information.

2. **Prevention of Symmetry**: By using round constants C_t, the key schedule eliminates symmetry between the round keys. If the round keys were symmetric, attackers could exploit this symmetry in differential or linear cryptanalysis.

3. **Resistance to Related-Key Attacks**: The key schedule ensures that the round keys are sufficiently different from one another, which makes related-key attacks (where an attacker uses pairs of keys with known relationships) more difficult.

By applying these transformations, the key schedule makes it computationally difficult for an attacker to exploit any regularities or patterns that could reveal information about the secret key.

3.4 Pseudocode for Key Schedule

The following pseudocode describes the process of key expansion, where the round keys are derived from the initial key:

```
Input: Key K (128-bit)
Output: Round Keys k_0, k_1, ..., k_8

1. Split Key K into four 32-bit words: w_0, w_1, w_2, w_3
2. Initialize round constants: C_0, C_1, ..., C_7
3. for round t = 1 to 8:
   4. Rotate the previous round key: w_t = rotate(w_{t-1})
   5. XOR with round constant: w_t = w_t     C_t
   6. Store the derived round key: k_t = w_t
4. Output the round keys k_0, k_1, ..., k_8
```

Listing 2: Pseudocode for Key Scheduling

In this pseudocode, the round constants C_t are defined iteratively, and each round key k_t is derived by applying the rotation and XOR operations. The key schedule ensures that the round keys are distributed across the rounds and prevents any obvious patterns from emerging.

3.5 Inverse Key Schedule

The inverse key schedule is designed to allow the decryption process to be carried out efficiently. The inverse of each round key can be derived using the same transformations as in the key expansion, but in reverse order. This symmetric structure allows for a consistent decryption process and ensures that the same round keys are used in reverse during the decryption rounds.

The inverse transformation for each round key is as follows:

$$k_t^{-1} = \text{rotation}^{-1}(k_t) \oplus C_t$$

This allows the same transformations to be applied during decryption but in reverse order to retrieve the original plaintext.

—

4 S-box Analysis

The S-box in the SQUARE block cipher plays a crucial role in providing nonlinearity to the cipher, which is essential for its security. The design and analysis of the S-box ensure that it resists both differential and linear cryptanalysis. In this section, we analyze the S-box used in SQUARE, focusing on its construction, resistance to attacks, and the properties that make it a strong cryptographic tool.

4.1 Design of the S-box

The S-box used in SQUARE is constructed over $GF(2^8)$, the finite field of 8-bit elements. It is designed to provide high nonlinearity, low differential uniformity, and resistance to algebraic attacks. The construction of the S-box involves two main steps:

1. **Multiplicative Inverse in** $GF(2^8)$: The first step is to take the multiplicative inverse of each byte in $GF(2^8)$, excluding 0, since 0 does not have an inverse. This step ensures that the S-box operates on the inverse of each byte, which is crucial for providing resistance to differential cryptanalysis.

2. **Affine Transformation**: After the multiplicative inverse, the output is transformed using an affine transformation. The affine transformation is defined by the matrix A and the constant vector b over $GF(2)$. The transformation is designed to increase the confusion in the cipher and prevent simple algebraic representations that could lead to cryptanalysis.

$$S(x) = \text{Affine}(\text{Inverse}(x))$$

Where the affine transformation is applied over $GF(2)$, making the overall mapping non-linear and difficult to reverse algebraically.

$$S(x) = A \cdot x + b \mod 2$$

Where: - A is the affine matrix over $GF(2)$, - b is the constant vector, - x is the 8-bit input vector (representing the byte).

The combination of these two operations (inverse and affine transformation) ensures that the S-box is invertible, and its cryptographic strength is maintained.

4.2 Differential Distribution Table (DDT)

The Differential Distribution Table (DDT) is a tool used to analyze the S-box's resistance to differential cryptanalysis. It provides the probability of a specific input difference Δx leading to a particular output difference Δy after the substitution step.

The DDT for an S-box is defined as:

$$E_{\Delta x, \Delta y} = \#\{x \mid S(x \oplus \Delta x) \oplus S(x) = \Delta y\}$$

Where: - Δx is the input difference, - Δy is the output difference, - # denotes the count of occurrences for the given difference pair.

The goal is to minimize the maximum value of $E_{\Delta x, \Delta y}$, as this would indicate that no differential trail has a high probability of occurring, which makes the S-box resistant to differential attacks.

For the SQUARE cipher, the S-box is designed such that its differential uniformity δ is minimized, ideally achieving a uniform distribution of output differences. The value δ

represents the maximum number of occurrences of any input/output difference pair and should be kept as low as possible for cryptographic security.

4.3 Linear Approximation Table (LAT)

The Linear Approximation Table (LAT) measures the correlation between a linear combination of input bits and output bits of the S-box. A low maximum bias in the LAT indicates that the S-box is resistant to linear cryptanalysis.

The LAT for an S-box is defined as:

$$\text{LAT}(a, b) = \sum_{x} (-1)^{a \cdot x \oplus b \cdot S(x)}$$

Where: - a is a vector representing the linear combination of input bits, - b is a vector representing the linear combination of output bits, - x is the input vector to the S-box, - $S(x)$ is the output of the S-box.

The goal of the LAT is to minimize the maximum bias, as this ensures that the S-box resists linear cryptanalysis. A low bias indicates that no linear approximation has a high correlation, thus enhancing the security of the cipher.

In the case of the SQUARE cipher, the S-box is designed such that the maximum bias $\text{LAT}(a, b)$ is typically on the order of 2^{-3}, which is considered low and ensures that the cipher is resistant to linear cryptanalysis.

4.4 Comparison with AES S-box

The SQUARE S-box shares many design goals with the AES S-box, such as ensuring high nonlinearity and low differential uniformity to resist both differential and linear cryptanalysis. However, there are differences in the specific construction methods and the mathematical basis used for the S-box.

Both S-boxes are designed to minimize the probability of certain differential trails and reduce the maximum correlation in linear approximations, making them both resistant to cryptanalytic attacks. However, the SQUARE S-box uses a different approach for creating its nonlinear mapping, which contributes to the overall structure and security of the cipher.

4.5 Visualization of the S-box

The S-box used in the SQUARE cipher can be visualized as a lookup table that maps each 8-bit input to a corresponding 8-bit output. The table is designed to have good cryptographic properties, with a minimal maximum differential uniformity and linear bias.

```
sbox = [
    0xc6, 0x37, 0x87, 0x47, 0xdf, 0x46, 0x06, 0xac, 0xf3, 0xe0, 0x86, 0x42, 0x1f, 0x8d, 0x4a, 0x97,
    0x5c, 0xd8, 0x6c, 0x27, 0x5f, 0x65, 0x84, 0xff, 0x2a, 0xbd, 0xda, 0x0a, 0x39, 0xba, 0xd7, 0xfc,
    0x8b, 0x2f, 0xc9, 0x92, 0x93, 0x03, 0x8f, 0x3c, 0xb3, 0xaa, 0xae, 0xef, 0xe7, 0x7d, 0xe3, 0xa1,
    0xb0, 0x8c, 0xc2, 0xcc, 0x71, 0x99, 0xa0, 0x59, 0x80, 0xd1, 0xf8, 0xde, 0x4e, 0x82, 0xdb, 0xa7,
    0x60, 0xc8, 0x32, 0x51, 0x41, 0x16, 0x55, 0xfa, 0xd5, 0x43, 0x9d, 0xcb, 0x62, 0xce, 0x02, 0xb8,
    0xc5, 0xed, 0xf0, 0x2e, 0xf2, 0x3f, 0xeb, 0x45, 0x56, 0x4c, 0x1b, 0x63, 0x54, 0x34, 0x75, 0x0c,
    0xfd, 0x0e, 0x5a, 0x4f, 0xc4, 0x24, 0xc3, 0xa8, 0xa4, 0x6f, 0xd0, 0x07, 0xf5, 0x33, 0x09, 0x7a,
    0xe5, 0xca, 0xf4, 0x08, 0xd9, 0x29, 0x73, 0xaf, 0x3b, 0x9b, 0x5d, 0xe2, 0xf1, 0x0f, 0xcf, 0xdd,
    0x2c, 0x30, 0xc1, 0x3e, 0x05, 0x89, 0xb4, 0x81, 0xbc, 0x8a, 0x17, 0x23, 0xb6, 0x25, 0x61, 0xc7,
    0xf6, 0xe8, 0x04, 0x3d, 0xd2, 0x52, 0xf9, 0x78, 0x94, 0x1e, 0x7b, 0xb1, 0x1d, 0x15, 0x40, 0x4d,
    0xfe, 0xd3, 0x53, 0x50, 0x64, 0x90, 0xb2, 0x35, 0xdc, 0xcd, 0x3a, 0xd6, 0xe9, 0xa9, 0xbe, 0x67,
    0x8e, 0x7c, 0x83, 0x26, 0x28, 0xad, 0x14, 0x6a, 0x36, 0x95, 0xbf, 0x5e, 0xa6, 0x57, 0x1a, 0x70,
    0x5b, 0x77, 0xa2, 0x12, 0x31, 0x9a, 0xbb, 0x9c, 0x7e, 0x2d, 0xb7, 0x01, 0x44, 0x2b, 0x48, 0x58,
    0xf7, 0x13, 0xab, 0x96, 0x74, 0xc0, 0x9f, 0x10, 0xe6, 0xa3, 0x85, 0x6b, 0x98, 0xec, 0x21, 0x19,
    0xee, 0x7f, 0x79, 0xe1, 0x66, 0x6d, 0x18, 0xb9, 0x49, 0x11, 0x88, 0x6e, 0x1c, 0xa5, 0x72, 0x0d,
    0x38, 0xea, 0x68, 0x20, 0x0b, 0x9e, 0xd4, 0x76, 0xe4, 0x69, 0x22, 0x00, 0xfb, 0xb5, 0x4b, 0x91
]
```

Figure 3: The S-box used in the SQUARE cipher.

The diagram above shows how each byte of the state matrix is substituted using the S-box during the encryption process. This substitution is what provides nonlinearity and confusion in the cipher.

4.6 Conclusion

The S-box is a vital component in the SQUARE block cipher, ensuring that the cipher provides the necessary nonlinearity to resist differential and linear cryptanalysis. Its design minimizes differential uniformity and linear bias, which are critical properties for the cipher's security. By using a combination of multiplicative inversion and affine transformation, the S-box offers strong resistance to cryptanalysis while maintaining efficiency for both hardware and software implementations.

—

5 Cryptographic Properties

The SQUARE block cipher employs several cryptographic properties to ensure its security. These include confusion and diffusion achieved through its round transformations, the Wide Trail Design Strategy, and well-defined security margins. This section discusses these properties in detail.

5.1 Confusion and Diffusion in SQUARE

Confusion and diffusion are fundamental principles that enhance the security of block ciphers.

- **Confusion**: SQUARE achieves confusion through the use of a nonlinear S-box substitution (γ), which replaces each byte with a non-linear transformation over $GF(2^8)$. The S-box is designed to have low differential uniformity ($\delta = 2^{-6}$) and high nonlinearity, ensuring resistance to differential and linear cryptanalysis.
- **Diffusion**: The linear transformation (θ) and byte permutation (π) provide strong diffusion. The linear transformation spreads the influence of each byte across the entire row of the state matrix, while the byte permutation reorders rows and columns to propagate this influence throughout the cipher. This ensures that small changes in the plaintext result in large, unpredictable changes in the ciphertext.

Together, confusion and diffusion ensure that SQUARE is resistant to cryptanalytic techniques that attempt to exploit patterns in the cipher's output.

5.2 Wide Trail Design Strategy

The Wide Trail Design Strategy, introduced in earlier cryptographic research, forms the foundation of SQUARE's resistance to differential and linear cryptanalysis. This strategy is implemented through a combination of nonlinear substitutions and linear transformations.

5.2.1 Active S-boxes and Trail Probability

The strategy focuses on eliminating high-probability differential trails and high-correlation linear trails by ensuring the presence of many **active S-boxes** in every round. An **active S-box** is one where either:

- A non-zero input difference propagates to a non-zero output difference (differential trails), or

- Non-zero linear input/output selection patterns result in non-zero correlation (linear trails).

The probability of a differential trail is approximated as:

$$P(\text{Trail}) = \prod_{\text{Active S-boxes}} P_{\text{Difference Propagation}}$$

Similarly, the correlation of a linear trail is computed as:

$$C(\text{Trail}) = \prod_{\text{Active S-boxes}} C_{\text{Input-Output Correlation}}$$

By choosing S-boxes with minimal differential uniformity and maximum nonlinearity, SQUARE ensures that these probabilities remain low, effectively resisting differential and linear cryptanalysis.

5.2.2 Role of Linear Transformations (μ) and Byte Permutation (π)

The linear transformation (θ) and byte permutation (π) are designed to:

- Spread differences across multiple active S-boxes over consecutive rounds.
- Ensure that no differential or linear trail involves only a few active S-boxes.

This approach ensures that every differential or linear trail involves a high number of active S-boxes, reducing the probability or correlation of such trails exponentially across multiple rounds.

5.3 Security Margins and Safety Rounds

SQUARE's design, with 8 rounds, provides a balanced trade-off between security and efficiency:

- **Differential and Linear Resistance**: With four active S-boxes per round and a low differential uniformity ($\delta = 2^{-6}$), the cipher maintains a strong resistance to both differential and linear cryptanalysis. A minimum of six rounds is considered sufficient for resistance against these attacks, providing two additional safety rounds for enhanced security.
- **Attack Complexity**: Advanced attacks, such as related-key boomerang and biclique cryptanalysis (discussed in Section 6), demonstrate the robustness of SQUARE's 8-round design. Extending the rounds further would significantly increase the attack complexity, but this is unnecessary for practical security under most scenarios.

—-

6 Cryptanalysis

6.1 Differential Cryptanalysis

SQUARE Cipher was designed keeping in mind that there exists great resistance towards differential cryptanalysis. Differential Cryptanalysis is used in order to check the propagation of a difference in the input bits all the way across the cipher's steps and the output also reflecting the difference.The probability associated with a differential characteristic is the probability that all intermediate difference patterns have the value specified in the above series. We call a differential characteristic a differential trail.

The probability associated with a differential trail can be approximated by the product of the difference propagations via DDT between every pair of subsequent rounds (which can be easily calculated). So, in general the probability that given an input difference a for input, the number of cipher rounds giving rise to some difference pattern b is equal to the sum of all such probabilistic trails starting with a and ending with b. The propagation of such difference from a to b is known as a differential.
During the study of this cipher therefore, we got to know by extensive testing that the SBox Sγ for this cipher is generated such that it holds up to differential attacks. What this means is that it is chosen so as to minimize the maximum probability of differential trails. We therefore tried to do cryptanalysis over four rounds of the cipher and found out that even at its maximum, the difference was propagated correctly only with a probability of at max 2^{-24} which makes it infeasible to do differential cryptanalysis on this cipher. This is due to high levels of diffusion and is also given in the paper as a design choice for resistance against differential and linear cryptanalysis by choosing the wide trail design strategy.
The code attached tries to generate such trails along with the probabilities of each of them. Changing the filter for probability shows that due to really low probabilities however, even the highest probability takes exponentially longer time for trail generation due to the SBox being really large leading to an even larger DDT having at max value of 4.

6.2 Integral Cryptanalysis

Integral cryptanalysis exploits the concept of balancedness in specific properties of a block cipher. For SQUARE, a 4-round integral distinguisher was developed to analyze the propagation of bytes through rounds, demonstrating how balance occurs in the 4th round after the θ operation.

6.2.1 Properties of the Integral Attack

The integral attack leverages the following properties:

- **All** A: The input set contains all possible values.
- **Constant** C: Specific bytes remain constant throughout the rounds.
- **Balanced** B: After 4 rounds, the XOR sum of certain bytes across all inputs equals 0.

6.2.2 Attack Procedure

The procedure for the integral cryptanalysis attack is as follows:

1. **Guess a Byte from** k_4: Choose a candidate byte k_4^{ij} from the 4th round key.
2. **Reverse Calculation**: Compute the value $S_{\theta 4}^{ij}$ by applying the inverse S-box and the θ transformation.
3. **Verification**: Check the XOR sum of all 256 inputs for the guessed subkey. If the result is non-zero, discard the guess.
4. **Key Candidate Selection**: The probability of a random 8-bit XOR sum being zero is 2^{-8}. With 2^8 guesses, only one candidate key is expected to pass.
5. **Practical Iterations**: For increased accuracy, sets of 2^k guesses are used to ensure higher probabilities of success.

6.2.3 Extended Rounds

The integral attack can be extended from the initial to the final rounds, increasing its complexity. For a 6-round attack, the **DTM complexity** is estimated as $(2^{32}, 2^{72}, 2^{32})$, indicating the required data, time, and memory resources.

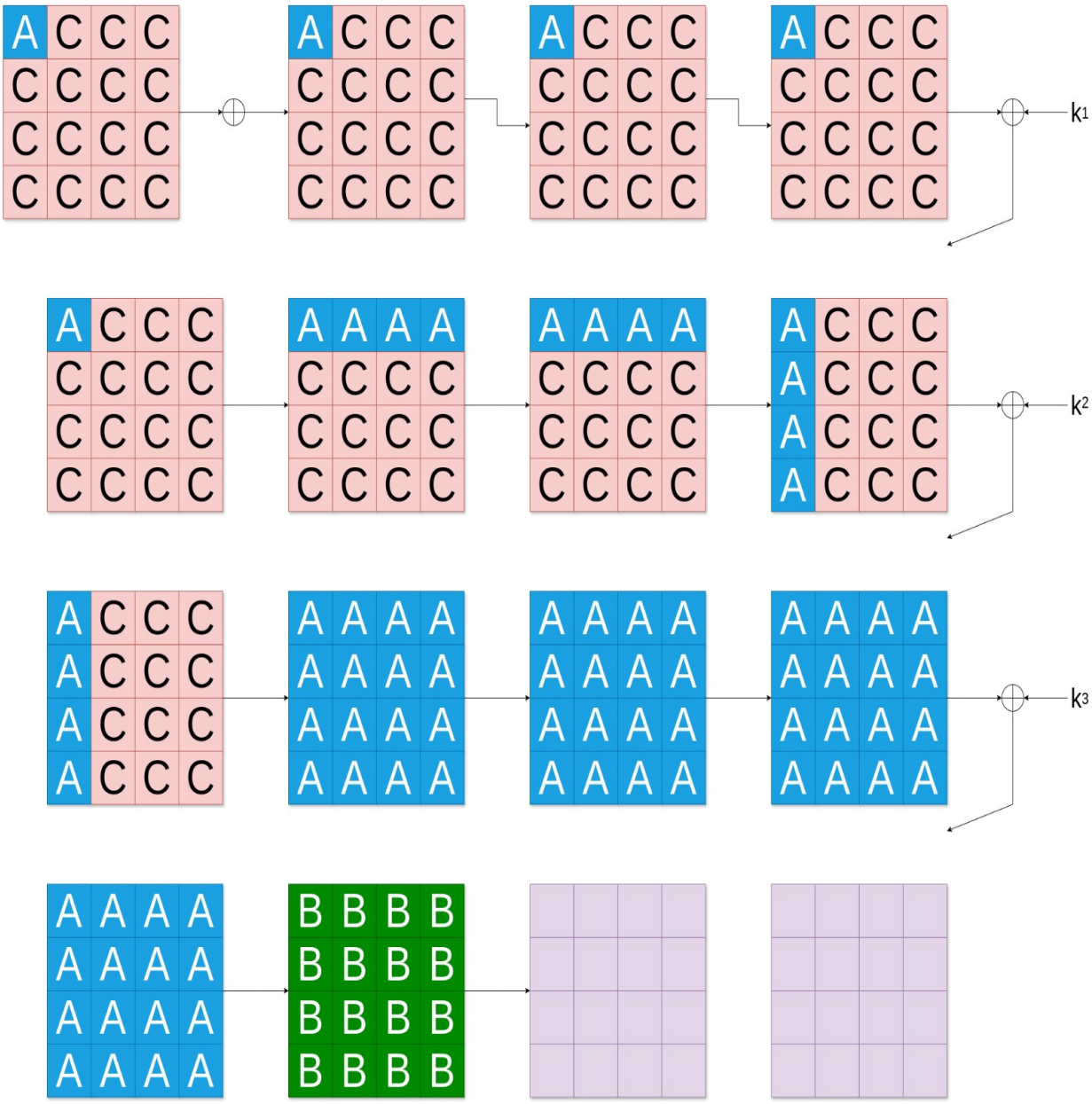

Figure 4: Integral Cryptoanalysis.

6.3 Related-Key Boomerang Attack

The related-key boomerang attack is a cryptanalytic technique that exploits weaknesses in both the cipher's round transformations and its key schedule. In SQUARE, this attack

takes advantage of the affine nature of the key schedule, which simplifies related-key differential analysis.

6.3.1 Key Insights

- **Distinguisher**: A 7-round related-key boomerang distinguisher is constructed with a probability of 2^{-119}. This distinguisher leverages **local collisions** and **ladder switch techniques** to amplify the probability of transitions across rounds.
- **Key Recovery**: By adding one additional round to the distinguisher, the attack targets the full 8-round SQUARE cipher and retrieves 16 bits of the round key.
- **Complexity**: The attack requires 2^{213} plaintext-ciphertext pairs (data complexity) and 2^{36} encryption operations (time complexity) to successfully recover the targeted key bits.

The related-key boomerang attack demonstrates the effectiveness of combining **related-key differentials** with **boomerang techniques** to exploit structural weaknesses in SQUARE.

6.4 Biclique Cryptanalysis

Biclique cryptanalysis is an advanced technique inspired by its successful application to AES. The biclique structure divides the cipher into two overlapping subspaces, allowing efficient key recovery through a combination of **differential** and **meet-in-the-middle techniques**.

6.4.1 Key Insights

- **Structure**: The attack constructs independent bicliques for overlapping parts of the cipher, reducing the number of key candidates by efficiently pruning the search space.
- **Complexity**: For the 8-round SQUARE cipher, the **DTM complexity** is:
  - **Data Complexity**: 2^{48},
  - **Time Complexity**: 2^{126},
  - **Memory Complexity**: 2^{16}.
- **Methodology**: By aligning bicliques with key-dependent structures in the cipher, the attack effectively recovers the full key with significantly fewer operations than exhaustive search.

Biclique cryptanalysis highlights the importance of optimizing the biclique structure to exploit specific design features of SQUARE, achieving practical attacks against its full-round implementation.

—

7 Comparison with AES

The Advanced Encryption Standard (AES) is a widely used block cipher that shares several design principles with SQUARE. AES was directly inspired by SQUARE's structural components and cryptographic strategies. This section highlights the similarities and differences between the two ciphers, focusing on their design, performance, and security.

7.1 Design Principles

Both SQUARE and AES are Substitution-Permutation Network (SPN) block ciphers, but they differ in their specific design choices and objectives.

- **Block Size and Key Size**:
  - SQUARE operates on a fixed block size of 128 bits and supports a key size of 128 bits.
  - AES also uses a block size of 128 bits but supports variable key sizes: 128, 192, and 256 bits, providing greater flexibility.
- **Round Transformations**:
  - Both ciphers utilize four main transformations per round: nonlinear substitution (S-box), linear transformation, permutation, and round key addition.
  - AES replaces SQUARE's θ transformation with the **MixColumns** operation, which uses a fixed matrix multiplication over $GF(2^8)$.
  - SQUARE uses the byte permutation (π) to reorder rows and columns, whereas AES employs **ShiftRows**, which cyclically shifts rows without column reordering.
- **Key Schedule**:
  - SQUARE's key schedule is simpler and entirely affine, making it efficient but more susceptible to related-key attacks.
  - AES incorporates non-linear key scheduling with additional round constants to enhance security against related-key attacks.

7.2 Performance Metrics

Performance differences arise due to variations in design complexity and optimization for specific platforms.

- **Computational Efficiency**:
  - SQUARE's simplicity in transformations and key schedule makes it computationally efficient for resource-limited environments.
  - AES, while more complex, has been heavily optimized for modern CPUs and supports hardware acceleration through the AES-NI instruction set.
- **Memory Usage**:
  - SQUARE requires minimal memory for lookup tables due to its lightweight structure.
  - AES uses additional memory for its larger S-box and MixColumns operations, which include precomputed tables for efficient implementation.
- **Hardware Implementation**:
  - SQUARE's structure is simpler to implement in hardware, making it suitable for embedded systems and smart cards.
  - AES is designed for both hardware and software environments, with dedicated hardware modules for encryption and decryption.

7.3 Cryptographic Strength

AES was designed to provide enhanced security over SQUARE by addressing vulnerabilities in SQUARE's structure and increasing resistance to attacks.

- **Differential and Linear Cryptanalysis**:
  - SQUARE is resistant to differential and linear cryptanalysis, but its security margin is narrower due to its 8-round design.
  - AES extends this margin with 10, 12, and 14 rounds for 128, 192, and 256-bit keys, respectively, making it more resistant to these attacks.
- **Advanced Attacks**:
  - SQUARE's affine key schedule is vulnerable to **related-key boomerang attacks** and other advanced techniques, as discussed in Section 6.
  - AES's non-linear key schedule and increased round count make it more robust against advanced cryptanalytic techniques, including biclique cryptanalysis.
- **Wide Trail Design Strategy**:
  - Both ciphers use the Wide Trail Design Strategy to minimize high-probability differential and linear trails. However, AES further enhances this strategy with its MixColumns and ShiftRows operations.

7.4 Summary of Comparison

Table 1 provides a summary of the comparison between SQUARE and AES.

Table 1: Comparison of SQUARE and AES.

Feature	**SQUARE**	**AES**
Block Size	128 bits	128 bits
Key Size	128 bits	128, 192, 256 bits
Rounds	8	10, 12, 14
Linear Transformation	θ	MixColumns
Permutation	Byte Permutation	ShiftRows
Key Schedule Complexity	Simple, Fully Affine	Non-linear
Differential Uniformity (δ)	2^{-6}	2^{-6}
Maximum LAT Bias	2^{-3}	2^{-4}
Hardware Optimization	High	Very High
Software Optimization	Moderate	Excellent

8 Practical Implementations

The practical implementation of the SQUARE block cipher demonstrates its versatility and efficiency across various computational platforms. This section explores the cipher's performance in software and hardware environments, emphasizing its suitability for resource-constrained systems and cryptographic applications.

8.1 Software Implementation

SQUARE is designed to be computationally efficient, making it a strong candidate for software-based cryptographic solutions. Its lightweight structure and simple transformations allow it to be implemented effectively on a wide range of processors.

- **8-bit Processors**: SQUARE's operations, such as the θ transformation and byte permutation (π), can be efficiently implemented on 8-bit microcontrollers. Its use of a single 8-bit S-box reduces memory requirements, making it suitable for embedded systems with limited resources.
- **32-bit Processors**: On 32-bit processors, SQUARE benefits from parallel processing capabilities, as the 4x4 state matrix can be represented using four 32-bit words. This parallelism significantly improves encryption speed while maintaining a small code footprint.
- **Performance Metrics**: Software benchmarks indicate that SQUARE achieves high throughput with minimal memory usage. Its simplicity enables consistent performance across various platforms, making it an ideal choice for lightweight cryptographic applications.

8.2 Hardware Implementation

The design simplicity of SQUARE facilitates its hardware implementation, offering excellent performance for applications requiring high-speed encryption or resource-constrained environments.

- **Compact Design**: SQUARE's structure allows for compact circuit design. The S-box can be implemented using small lookup tables, while the linear transformation (θ) can be efficiently realized with XOR gates and shift registers.
- **Low Power Consumption**: The minimal complexity of SQUARE's operations ensures that it consumes less power, making it suitable for battery-operated devices such as smart cards, IoT devices, and RFID systems.
- **Hardware Throughput**: Due to its efficient round transformations, SQUARE achieves high throughput in hardware. It is particularly effective for pipelined architectures, where each transformation can be executed in a separate pipeline stage.
- **Implementation Feasibility**: FPGA and ASIC implementations of SQUARE demonstrate its adaptability to diverse hardware platforms. Its small gate count and low latency make it a practical choice for cryptographic hardware.

8.3 Applications

The efficiency of SQUARE in both software and hardware environments makes it suitable for a wide range of applications:

- **Embedded Systems**: Due to its lightweight design, SQUARE is ideal for embedded systems, including microcontrollers and low-power devices.
- **Secure Communication**: SQUARE can be integrated into protocols requiring lightweight encryption, such as secure messaging and file encryption in resource-constrained networks.

- **IoT Devices**: With its low power consumption and compact design, SQUARE is suitable for securing IoT communications where energy efficiency is critical.

- **Smart Cards and RFID**: Hardware implementations of SQUARE are particularly effective for smart card and RFID systems, which demand small, efficient cryptographic cores.

- **NFT Marketplace**: One notable application of SQUARE's encryption is in the NFT marketplace we have developed. The marketplace leverages secure and efficient encryption for transactions and storage, ensuring both buyer and seller data is protected. The system uses blockchain principles and SQUARE's encryption to maintain confidentiality and integrity throughout the buying, selling, and storing of digital assets like art NFTs.

  The marketplace's design allows users to securely buy, sell, and store NFTs using a simple yet effective interface. Each transaction is encrypted, and user wallets are managed securely, utilizing SQUARE's lightweight encryption to protect sensitive financial information.

8.3.1 Step-by-Step Process of NFT Marketplace

The NFT marketplace I developed operates with the following steps:

- **Login/Registration**: - Users start by either logging in with existing credentials or registering for a new account. The registration process involves providing a secure email and password, and users can also link their wallets for seamless transactions. - After registration, the password is encrypted by the Square Cipher and then JWT.

- **Wallet Integration**: - Once the user is created a wallet is automatically made for that particular user and details are stored in encrypted format in the MongoDB.

- **Create NFTs**: - Users can create a wide variety of digital art pieces listed as NFTs. Once the details of the artwork is given then a encrypted metadata is created using **Square Cipher** containing the details of the users

- **Browse and Buy NFTs**: - Users can browse a wide variety of digital art pieces listed as NFTs. Each artwork displays metadata, including the price and details, along with an option to purchase. - When a user decides to buy an NFT, the wallet balance is checked to ensure sufficient funds are available. Once the purchase is confirmed, the NFT is transferred to the user's account and the encrypted metadata is changed to the buyers data.

- **View Owned NFTs**: - After logging in, users can view a list of NFTs they own. Each NFT in the collection is displayed with the artwork image, details, and purchase history. - This feature is powered by secure metadata storage, ensuring that ownership records are immutable and safe from tampering.

8.4 Summary of Implementation Features

Table 2 provides a summary of SQUARE's implementation features in software and hardware.

—-

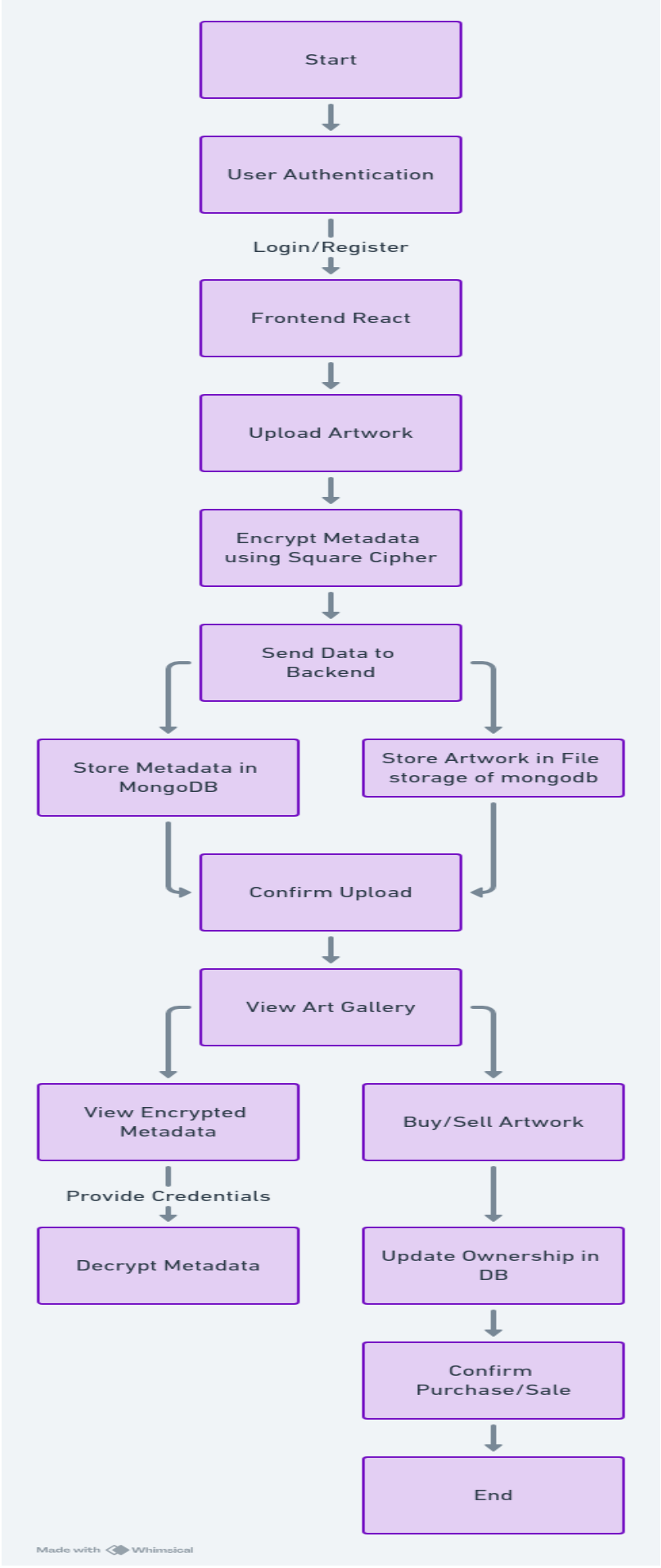
Start
User Authentication
Login/Register
Frontend React
Upload Artwork
Encrypt Metadata using Square Cipher
Send Data to Backend
Store Metadata in MongoDB
Store Artwork in File storage of mongodb
Confirm Upload
View Art Gallery
View Encrypted Metadata
Buy/Sell Artwork
Provide Credentials
Decrypt Metadata
Update Ownership in DB
Confirm Purchase/Sale
End
Made with Whimsical

Table 2: Summary of SQUARE's Practical Implementation Features.

Feature	**Software Implementation**	**Hardware Implementation**
Memory Usage	Low	Very Low
Processing Speed	Moderate to High	High
Power Consumption	Moderate	Low
Suitability for Embedded Systems	High	Very High
Hardware Efficiency	N/A	Excellent
Implementation Complexity	Low	Low

9 Conclusion

The SQUARE block cipher, proposed as a precursor to AES, represents a significant milestone in the evolution of cryptographic algorithms. This paper has provided an in-depth analysis of its structure, cryptographic properties, and vulnerability to various cryptanalytic techniques.

9.1 Summary of Findings

The study highlights several critical aspects of SQUARE:

- **Structural Design**: SQUARE employs a Substitution-Permutation Network (SPN) structure with a fixed block size of 128 bits and a key size of 128 bits. Its innovative transformations, including the linear transformation (θ), byte permutation (π), and round key addition (σ), ensure high diffusion and confusion.
- **Cryptographic Strength**: The cipher demonstrates robust resistance to differential and linear cryptanalysis, leveraging its Wide Trail Design Strategy and S-box properties. However, its simple affine key schedule makes it susceptible to advanced attacks like related-key boomerang and biclique cryptanalysis.
- **Cryptanalysis**: The analysis of integral cryptanalysis revealed that the cipher achieves balanced outputs after four rounds, highlighting its strong diffusion properties. Advanced attacks, such as the related-key boomerang and biclique cryptanalysis, exploit structural weaknesses, offering practical insights into the cipher's security margin.
- **Comparison with AES**: While SQUARE influenced the design of AES, the latter addresses several limitations of SQUARE, including enhanced key scheduling and higher security margins. AES also extends the block cipher's applicability with variable key lengths and improved optimization for modern hardware and software.
- **Practical Implementations**: SQUARE's lightweight structure makes it ideal for resource-constrained environments, such as embedded systems and IoT devices. Its compact hardware implementation and moderate computational requirements ensure its feasibility across diverse applications.

9.2 Legacy and Influence

SQUARE's design principles, particularly the Wide Trail Design Strategy and its SPN structure, significantly influenced the development of AES. The cipher's introduction of balanced nonlinear transformations and efficient diffusion operations paved the way for modern block cipher designs. Despite its reduced rounds and vulnerabilities to certain advanced attacks, SQUARE remains an academically important cipher, providing valuable lessons for future cryptographic innovations.

9.3 Future Directions

While SQUARE is no longer recommended for practical applications, its principles can inspire the development of lightweight cryptographic algorithms. Future research could explore:

- **Enhanced Key Scheduling**: Incorporating non-linear elements into the key schedule to mitigate vulnerabilities to related-key attacks.
- **Increased Round Count**: Extending the number of rounds to improve security margins without compromising efficiency.
- **Applications in Lightweight Cryptography**: Leveraging SQUARE's efficiency for developing cryptographic primitives tailored to IoT and low-power environments.

9.4 Final Remarks

In conclusion, the SQUARE block cipher exemplifies a pivotal transition in cryptographic design, bridging theoretical innovation with practical applicability. Its influence on AES underscores its historical and academic significance. While its practical deployment has been superseded by more robust algorithms, the principles underlying SQUARE continue to inspire advancements in cryptography.

References

@articlearticle, author = Daemen, Joan and Knudsen, Lars and Rijmen, Vincent, year = 1998, month = 10, pages = , title = The block cipher Square, volume = 1267, isbn = 978-3-540-63247-4, journal = Lecture Notes in Computer Science, doi = 10.1007/BFb0052343