# Cryptography with Information Theory

**DSN4092 - CAPSTONE PROJECT PHASE-II**

**Phase – II Report**

***Submitted by***


| Name | Reg.No |
|---|---|
| Karan Kumar Chauhan | 21BCE10243 |
| Abhishek Kumar | 21BCE10084 |
| Deep Radadiya | 21BCE11343 |
| Vikash Kumar Sinha | 21BCE11500 |
| Amul Gupta | 21BCE11606 |


***in partial fulfillment of the requirements for the degree of***

***Bachelor of Engineering and Technology***


VIT®
BHOPAL
www.vitbhopal.ac.in


**VIT Bhopal University**
**Bhopal**
**Madhya Pradesh**


**April, 2025**


VIT®
BHOPAL
www.vitbhopal.ac.in

## Bonafide Certificate

Certified that this project report titled **Cryptography with Information Theory** is the bonafide work of **21BCE10243 Karan Kumar Chauhan; 21BCE10084 Abhishek Kumar; 21BCE11343 Deep Radadiya; 21BCE11500 Vikash Kumar Sinha; 21BCE11606 Amul Gupta** have carried out the project work under my supervision.

This project report (DSN4092 - Capstone Project Phase-II) is submitted for the Project Viva-Voce examination held between 01st April – 07th April 2025.

**Supervisor**
**Dr. Mohammad Sultan Alam**
**Assistant Professor**
**School of Computer Science and Engineering**
**VIT BHOPAL UNIVERSITY**

## Content

# CHAPTER 1: INTRODUCTION

In an era where digital communication is the backbone of modern society, ensuring the confidentiality, integrity, and authenticity of information has never been more critical. As cyber threats continue to evolve, robust encryption mechanisms are essential to safeguarding sensitive data against unauthorized access and malicious attacks. Cryptographic algorithms form the foundation of secure communication systems, and their effectiveness hinges on the strength and unpredictability of the cryptographic keys they generate.

One of the most fundamental properties that determine key strength is randomness, as predictable keys can be exploited through cryptanalysis, leading to security breaches. Shannon entropy, a widely accepted metric, quantifies the level of uncertainty in a system and serves as a key indicator of randomness. Higher entropy values signify stronger, more unpredictable keys that can withstand brute-force and statistical attacks, reinforcing their importance in encryption security.

This study presents a comparative analysis of key randomness and entropy levels in widely used symmetric and asymmetric cryptographic algorithms. The symmetric encryption methods analyzed include AES (Advanced Encryption Standard), 3DES (Triple DES), DES (Data Encryption Standard), and Blowfish, while the asymmetric methods evaluated encompass RSA (Rivest-Shamir-Adleman), DSA (Digital Signature Algorithm), and ECC (Elliptic Curve Cryptography). These algorithms are assessed based on their key randomness, entropy per byte, and computational efficiency, providing insights into their effectiveness for securing digital communications.

Findings from this research highlight key trade-offs between entropy, key size, and computational efficiency. The study underscores the significance of selecting encryption algorithms based on their entropy characteristics to ensure optimal security across diverse applications, including cloud computing, IoT devices, mobile security, and enterprise systems. By analyzing the interplay between randomness and encryption strength, this research contributes to the ongoing efforts to enhance cryptographic security in the face of emerging cyber threats.

## 1.1 Motivation

In an era where cyber threats are growing in sophistication and frequency, ensuring the security of cryptographic systems is of paramount importance. The foundation of encryption lies in the strength of its key generation process, which directly impacts its resistance against brute-force and cryptanalytic attacks. However, not all cryptographic algorithms generate keys with the same level of randomness, leading to variations in security effectiveness.

Entropy, a measure of unpredictability, plays a critical role in assessing the robustness of cryptographic keys. A higher entropy value signifies greater randomness, making it exponentially harder for adversaries to exploit patterns. Conversely, low entropy can introduce vulnerabilities, potentially compromising sensitive data. Given the increasing computational power available to attackers,

understanding how different encryption algorithms handle entropy is essential for making informed security decisions.

This research investigates the entropy characteristics of commonly used symmetric and asymmetric encryption algorithms to assess their suitability for modern security applications. By analyzing entropy per byte and key randomness, this study aims to provide valuable insights into the trade-offs between security strength, computational efficiency, and practical implementation constraints.

### 1.2 Objective

The primary objective of this research is to analyze and compare the entropy levels and key generation mechanisms of widely used symmetric and asymmetric cryptographic algorithms. Specifically, this study aims to:

1. **Evaluate Key Randomness:** Assess the level of unpredictability in cryptographic keys generated by AES, 3DES, DES, Blowfish, RSA, DSA, and ECC.
2. **Measure Entropy Levels:** Utilize Shannon entropy as a statistical metric to quantify the randomness and security strength of encryption keys.
3. **Compare Symmetric and Asymmetric Algorithms:** Examine the differences in entropy per byte, key size, and security efficiency between symmetric and asymmetric encryption techniques.
4. **Assess Security Implications:** Identify how entropy levels impact cryptographic strength and resistance to brute-force and cryptanalytic attacks.
5. **Provide Practical Insights:** Offer recommendations on selecting encryption algorithms based on their entropy properties, computational efficiency, and security requirements for diverse applications, including resource-constrained environments.

This study aims to enhance the understanding of key randomness in cryptographic systems, aiding in the selection of secure and efficient encryption techniques.

# CHAPTER 2: EXISTING WORK / LITERATURE REVIEW

The study of cryptographic key generation and entropy measurement has been a subject of extensive research due to its crucial role in ensuring secure communication. Various researchers have analyzed the randomness of key generation mechanisms in both symmetric and asymmetric cryptographic algorithms, focusing on entropy evaluation and its impact on security.

**Symmetric Encryption and Entropy Studies:**
Several studies have explored the entropy levels of symmetric encryption algorithms such as AES, 3DES, DES, and Blowfish. Research indicates that 3DES exhibits higher entropy due to its multi-layered encryption process, while AES and Blowfish also demonstrate strong randomness characteristics. However, DES has been identified as inadequate for modern security applications due to its lower entropy levels and susceptibility to brute-force attacks. Prior work highlights the necessity of longer key lengths and robust randomness sources to enhance symmetric encryption security.

**Asymmetric Cryptography and Entropy Analysis:**
The entropy characteristics of asymmetric encryption schemes, including RSA, DSA, and ECC, have been widely studied. Due to their reliance on mathematical structures, these algorithms generally exhibit lower entropy per byte. However, researchers have shown that larger key sizes compensate for this limitation, ensuring strong security. ECC has been recognized for its efficiency in providing robust encryption with smaller key sizes, making it a suitable choice for resource-constrained environments like IoT devices and mobile applications.

**Shannon Entropy in Cryptographic Analysis:**
Shannon entropy serves as a key metric for evaluating randomness in cryptographic systems. Existing literature emphasizes its significance in assessing the unpredictability of keys, with studies demonstrating that higher entropy values correspond to stronger security. Researchers have employed programmatic approaches to analyze entropy distribution across different cryptographic algorithms, reinforcing the importance of key randomness in preventing cryptanalytic attacks.

The study of cryptographic key generation and entropy measurement has been a subject of extensive research due to its crucial role in ensuring secure communication. Various researchers have analyzed the randomness of key generation mechanisms in both symmetric and asymmetric cryptographic algorithms, focusing on entropy evaluation and its impact on security.

**Symmetric Encryption and Entropy Studies:**
Several studies have explored the entropy levels of symmetric encryption algorithms such as AES, 3DES, DES, and Blowfish. Research indicates that 3DES exhibits higher entropy due to its multi-layered encryption process, while AES and Blowfish also demonstrate strong randomness characteristics. However, DES has been identified as inadequate for modern security applications due to its lower entropy levels and susceptibility to brute-force attacks. Prior work highlights the necessity of longer key lengths and robust randomness sources to enhance symmetric encryption security.

**Asymmetric Cryptography and Entropy Analysis:**
The entropy characteristics of asymmetric encryption schemes, including RSA, DSA, and ECC, have been widely studied. Due to their reliance on mathematical structures, these algorithms generally exhibit lower entropy per byte. However, researchers have shown that larger key sizes compensate for this limitation, ensuring strong security. ECC has been recognized for its efficiency in providing robust encryption with smaller key sizes, making it a suitable choice for resource-constrained environments like IoT devices and mobile applications.

**Shannon Entropy in Cryptographic Analysis:**
Shannon entropy serves as a key metric for evaluating randomness in cryptographic systems. Existing literature emphasizes its significance in assessing the unpredictability of keys, with studies demonstrating that higher entropy values correspond to stronger security. Researchers have employed programmatic approaches to analyze entropy distribution across different cryptographic algorithms, reinforcing the importance of key randomness in preventing cryptanalytic attacks.

**Comparative Studies on Cryptographic Security:**
Prior research has extensively compared symmetric and asymmetric encryption techniques, evaluating their trade-offs in terms of security, entropy, and computational efficiency. Findings suggest that while symmetric algorithms provide faster encryption and decryption, asymmetric encryption offers enhanced security through complex key structures and public-private key mechanisms. Recent advancements in cryptographic research continue to focus on optimizing entropy while balancing computational performance.

This literature review underscores the need for continuous evaluation of cryptographic key generation techniques. By building upon existing studies, this research aims to provide a comparative analysis of entropy levels in widely used encryption algorithms, contributing to the ongoing development of secure cryptographic standards.

# CHAPTER 3: FRONT-END, BACK-END, AND SYSTEM REQUIREMENTS

## FRONT END:

The system runs on Jupyter Notebook without a traditional graphical user interface (GUI). Instead, it provides an interactive environment where users execute Python scripts to perform encryption, decryption, and security analysis, enabling direct interaction with the underlying cryptographic processes.

**Key Features of the Frontend:**

1. **Command-Based Interaction:**
    - Users input plaintext messages and parameters directly in Jupyter Notebook cells.
    - Functions are executed step by step, allowing users to observe the encryption process in real time.
2. **Data Visualization Tools:**
    - **Entropy Graphs:** The system calculates and plots Shannon entropy values to analyze information security.
    - **Probability Distributions:** Visualization of key randomness using histograms.
    - **Encryption Process Demonstration:** Step-by-step representation of encoding, transmission, and decoding.
3. **Input & Output Handling:**
    - Text-based input allows users to enter plaintext messages for encryption.
    - Output displays encrypted ciphertext, decryption results, and security metrics.
    - Interactive debugging with immediate feedback from executed Python cells.

This structure ensures simplicity and flexibility, making it easy for users to experiment with secure communication concepts.

**BACKEND**:

The backend is responsible for performing all cryptographic operations, managing key exchanges, and ensuring secure communication. It leverages Python's computational power to implement security algorithms based on information theory.

**Key Functions of the Backend:**

1. **Mathematical Computation and Security Analysis:**
    - Shannon entropy calculations determine the randomness and security level of messages.
    - Probability and statistical functions are used for key generation and verification.
2. **Encryption & Decryption Algorithms:**
    - Implementation of custom encryption techniques based on information theory.
    - Use of Python's pycryptodome library for additional security mechanisms.
3. **Secure Key Exchange:**
    - Generation of secure keys using randomness-based cryptographic techniques.
    - Key distribution simulated within Jupyter Notebook without external dependencies.
4. **Data Transmission Simulation:**
    - The system simulates the secure transmission of encrypted messages.
    - Demonstrates potential attack scenarios and defenses.

The backend efficiently handles all security operations while ensuring high performance and modularity.

**SYSTEM REQUIREMENTS**

To run this system smoothly, certain software and hardware requirements must be met.

**Software Requirements:**

- **Operating System:** Windows, Linux, or macOS
- **Programming Language:** Python 3.x (Recommended: Python 3.8 or later)
- **Required Python Libraries:**
    - numpy – Numerical computations
    - scipy – Statistical functions
    - sympy – Symbolic mathematics
    - matplotlib – Data visualization
    - seaborn – Advanced plotting
    - pycryptodome – Cryptographic operations
    - pandas – Structured data handling

To install the required dependencies, run:

*python*
*CopyEdit*
*pip install numpy scipy sympy matplotlib pycryptodome pandas seaborn*

**Hardware Requirements:**

- **Minimum Configuration:**
  - Processor: Intel Core i3 / AMD equivalent
  - RAM: 4GB
  - Storage: 5GB free disk space
  - Internet: Required for installing dependencies

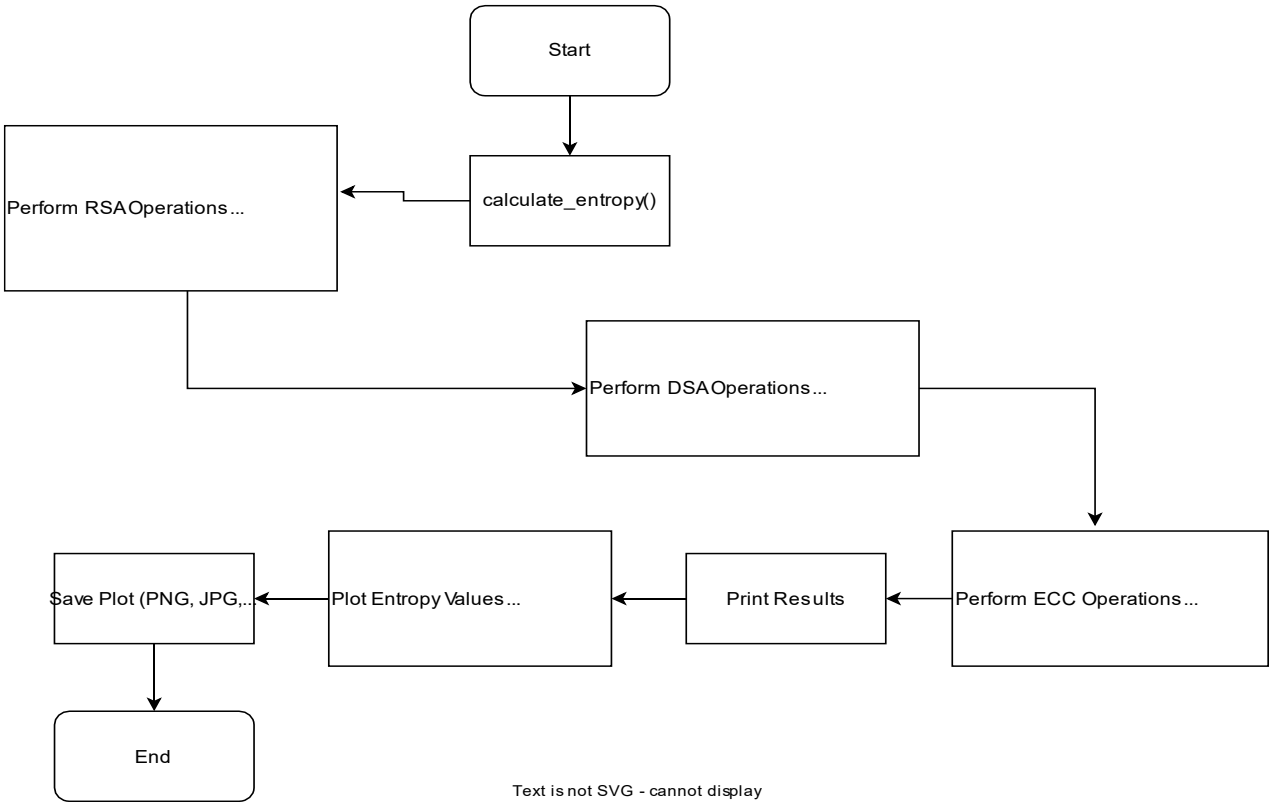
- **Recommended Configuration for Better Performance:**
  - Processor: Intel Core i5/i7 or AMD Ryzen
  - RAM: 8GB+
  - Storage: SSD for faster execution

This system is optimized to run on mid-range computers without requiring a dedicated server.

# CHAPTER 4: METHODOLOGY

## System Design / Architecture

The system follows Shannon's secure communication model, ensuring confidentiality and robustness in data transmission. The workflow consists of multiple stages, from message generation to encryption, transmission, decryption, and security evaluation. Below is a detailed step-by-step breakdown of the system's working:




### 1. Message Generation (Source)

- The system starts with the user inputting a plaintext message within a Jupyter Notebook cell.
- This plaintext is treated as a sequence of symbols, each with a defined probability, ensuring randomness in the data structure.
- The message can be manually entered or programmatically generated based on specific use cases.
- The entropy of the plaintext is calculated at this stage to assess its randomness and predictability.

**2. Encryption Process (Cipher System)**

- Once the plaintext is provided, the system applies an encryption algorithm based on Shannon's principles of **confusion** and **diffusion** to enhance security.
- A secret key, known only to the sender and the receiver, is used to encrypt the message.
- The encryption function scrambles the original message, converting it into an unintelligible ciphertext.
- The randomness and entropy of the ciphertext are analyzed, ensuring that it maintains a high level of security.

**3. Transmission Through a Noisy Channel**

- The encrypted message is then transmitted over a simulated noisy channel, mimicking real-world communication systems.
- In an insecure medium, the system assumes the presence of potential attackers who may attempt to intercept the message.
- To counteract such threats, Shannon's theorem ensures that even if an attacker captures the ciphertext, it remains computationally infeasible to retrieve the original plaintext without the decryption key.
- The noisy channel may introduce some distortions, which the system accounts for by implementing error detection mechanisms.

**4. Decryption Process at the Receiver's End**

- At the receiving end, the system applies a decryption function using the same secret key to retrieve the original message.
- The decryption process reverses the encryption steps, reconstructing the plaintext from the ciphertext.
- To verify the accuracy and integrity of the recovered message, error correction techniques may be employed.
- If an incorrect key is used, the decrypted output remains unintelligible, reinforcing the security model.

**5. Security Analysis and Entropy Measurement**

- To assess the robustness of encryption, the system calculates and visualizes Shannon entropy, measuring the uncertainty of the encrypted message.
- A higher entropy value indicates that the encryption is effective, making it harder for attackers to predict or decrypt the message.

- The system also evaluates probability distributions to ensure that the encrypted data does not reveal patterns that could be exploited.
- Graphical tools, such as histograms and entropy plots, help users analyze the security level of their encrypted messages.

This structured approach ensures that the system adheres to Shannon's **perfect secrecy** principle, making it resistant to brute-force attacks and statistical analysis.

**Working Principle**

The working principle of the system is based on secure information transmission using encryption, decryption, and entropy-based security analysis. The core working process follows these steps:

1. **Message Generation and Input Processing:**

- The user inputs a plaintext message which is the original data intended for secure communication.
- The system preprocesses the message to remove inconsistencies and ensure proper encoding.

2. **Key Generation and Encryption:**

- The system generates a secret key, ensuring high randomness and unpredictability.
- The plaintext is encrypted using symmetric or asymmetric cryptographic methods.
- The entropy of the generated ciphertext is analyzed to measure security effectiveness.

3. **Secure Transmission Through Noisy Channels:**

- The encrypted message is transmitted through a simulated noisy channel.
- The system assumes adversarial presence and tests resistance against interception.

4. **Decryption and Integrity Verification:**

- The receiver decrypts the message using the corresponding secret key.
- The decrypted output is compared with the original plaintext to verify transmission accuracy.

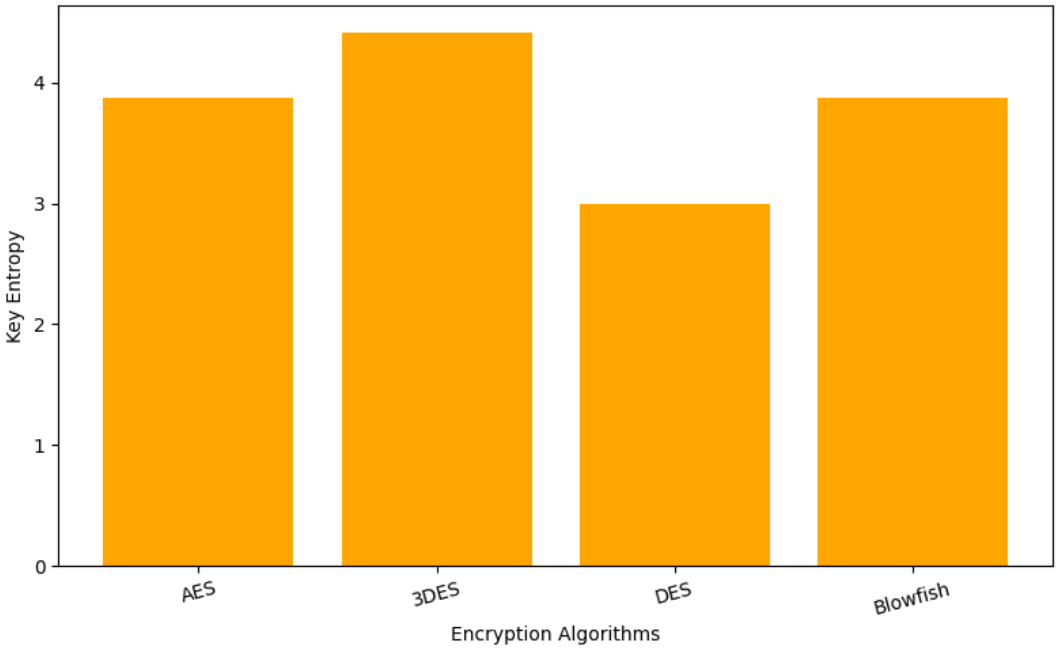
5. **Security Evaluation Using Entropy and Probability Distributions:**

- The randomness of encryption keys and ciphertext is visualized through entropy graphs and histograms.
- Higher entropy values indicate better security, preventing pattern recognition attacks.
- The system performs statistical analysis to identify vulnerabilities in encryption methods.

**Results and Discussion**

1. **Entropy Analysis Results:**
    - The computed entropy values indicate the randomness levels of different encryption algorithms.
    - AES and 3DES exhibit higher entropy, ensuring stronger security, while DES shows lower entropy, making it more vulnerable.
2. **Key Randomness and Probability Distributions:**
    - Histogram analysis confirms that secure key generation leads to an even probability distribution.
    - Weak keys show patterns that could make encryption susceptible to statistical attacks.
3. **Performance Evaluation:**
    - Encryption and decryption speed are analyzed for different algorithms.
    - Results highlight the trade-off between security strength and computational efficiency.
4. **Comparison with Traditional Cryptographic Methods:**
    - The system's implementation is compared with standard cryptographic practices to assess improvements in security and efficiency.

- **Symmetric Algorithms**




| Algorithm | *DES* | *3DES* | *AES* | *BOWLFISH* |
|---|---|---|---|---|
| Key | 3.000 | 4.303 | 4.0000 | 3.8750 |
| Entropy | 0 bits | 5 bits | bits | bits |

TABLE III

KEY ENTROPY TABLE FOR DIFFERENT SYMMETRIC ALGORITHMS

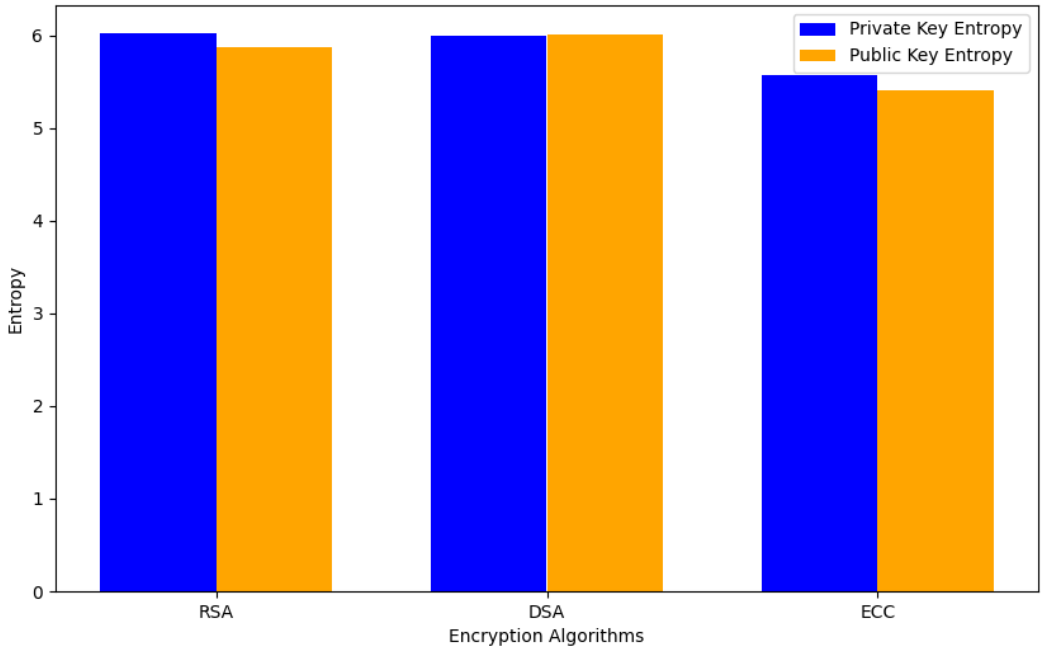
# Appendix:

Code:

```
import math
import matplotlib.pyplot as plt
from Crypto.PublicKey import RSA, DSA, ECC
from Crypto.Cipher import PKCS1_OAEP
from Crypto.Signature import DSS
from Crypto.Hash import SHA256

# Function to calculate entropy
def calculate_entropy(data):
    if isinstance(data, str):
        data = data.encode()  # Convert to bytes if not already
    length = len(data)
    freq = {byte: data.count(byte) / length for byte in set(data)}
    entropy = -sum(p * math.log2(p) for p in freq.values())
    return entropy

# RSA encryption and decryption
def rsa_encrypt_decrypt(text):
    key = RSA.generate(2048)
    public_key = key.publickey()
    cipher_rsa = PKCS1_OAEP.new(public_key)
    ct_bytes = cipher_rsa.encrypt(text.encode())
    cipher_rsa = PKCS1_OAEP.new(key)
    decrypted_text = cipher_rsa.decrypt(ct_bytes).decode()
    return ct_bytes, decrypted_text, key.export_key(), public_key.export_key()

# DSA signing and verification
def dsa_sign_verify(text):
    key = DSA.generate(2048)
    hash_obj = SHA256.new(text.encode())
    signer = DSS.new(key, 'fips-186-3')
    signature = signer.sign(hash_obj)
    return signature, key.export_key(), key.publickey().export_key()

# ECC signing and verification
def ecc_sign_verify(text):
    key = ECC.generate(curve='P-256')
    hash_obj = SHA256.new(text.encode())
    signer = DSS.new(key, 'fips-186-3')
    signature = signer.sign(hash_obj)
    return signature, key.export_key(format='PEM'),
key.public_key().export_key(format='PEM')

# Sample text
sample_text = "I love Cryptography"
```

```
# Perform RSA encryption
rsa_encrypted, rsa_decrypted, rsa_private_key, rsa_public_key =
rsa_encrypt_decrypt(sample_text)
rsa_private_entropy = calculate_entropy(rsa_private_key)
rsa_public_entropy = calculate_entropy(rsa_public_key)

# Perform DSA signing
dsa_signed, dsa_private_key, dsa_public_key = dsa_sign_verify(sample_text)
dsa_private_entropy = calculate_entropy(dsa_private_key)
dsa_public_entropy = calculate_entropy(dsa_public_key)

# Perform ECC signing
ecc_signed, ecc_private_key, ecc_public_key = ecc_sign_verify(sample_text)
ecc_private_entropy = calculate_entropy(ecc_private_key)
ecc_public_entropy = calculate_entropy(ecc_public_key)

# Display the results
for algo, priv_key, pub_key, enc_text, dec_text, priv_ent, pub_ent in zip(
    ["RSA", "DSA", "ECC"],
    [rsa_private_key, dsa_private_key, ecc_private_key],
    [rsa_public_key, dsa_public_key, ecc_public_key],
    [rsa_encrypted.hex(), dsa_signed.hex(), ecc_signed.hex()],
    [rsa_decrypted, "Not Applicable", "Not Applicable"],
    [rsa_private_entropy, dsa_private_entropy, ecc_private_entropy],
    [rsa_public_entropy, dsa_public_entropy, ecc_public_entropy]
):
    print(f"\n=== {algo} ===")
    print(f"Private Key: {priv_key.decode() if isinstance(priv_key, bytes) else
priv_key}")
    print(f"Public Key: {pub_key.decode() if isinstance(pub_key, bytes) else
pub_key}")
    print(f"Encrypted/Signed Text: {enc_text}")
    print(f"Decrypted/Verified Text: {dec_text}")
    print(f"Private Key Entropy: {priv_ent:.4f}")
    print(f"Public Key Entropy: {pub_ent:.4f}")

# Prepare data for plotting
algorithms = ["RSA", "DSA", "ECC"]
private_entropies = [rsa_private_entropy, dsa_private_entropy,
ecc_private_entropy]
public_entropies = [rsa_public_entropy, dsa_public_entropy, ecc_public_entropy]

# Plotting entropy values
plt.figure(figsize=(10, 6))



x = range(len(algorithms))
width = 0.35  # Width of the bars

plt.bar(x, private_entropies, width, label='Private Key Entropy', color='blue')
plt.bar([i + width for i in x], public_entropies, width, label='Public Key
Entropy', color='orange')

plt.xlabel('Encryption Algorithms')
plt.ylabel('Entropy')
plt.xticks([i + width / 2 for i in x], algorithms)
plt.legend()
plt.savefig("plot.png")  # Save as PNG
plt.savefig("plot.jpg", dpi=300)  # Save as JPG with higher resolution
plt.savefig("plot.pdf")
plt.show()
```

- **Asymmetric Algorithms**




| Algorithm | RSA | DSA | ECC |
|---|---|---|---|
| 2*Private Key | 6.019245 59942364 | 6.00108 0426728 4 | 5.615320622 83071 18 |
| 2*Public Key | 5.879935 20549828 | 6.01675 0633764 7 | 5.469152308 482562 302 |

TABLE IV
KEY ENTROPY TABLE FOR DIFFERENT ASYMMETRIC ALGORITHMS

## Code:

```
import math
import matplotlib.pyplot as plt
from Crypto.Cipher import AES, DES3, DES, Blowfish
from Crypto.Random import get_random_bytes
from Crypto.Util.Padding import pad, unpad

# Function to calculate entropy
def calculate_entropy(data):
    if isinstance(data, str):
        data = data.encode()  # Convert to bytes if not already
    length = len(data)
    freq = {byte: data.count(byte) / length for byte in set(data)}
    entropy = -sum(p * math.log2(p) for p in freq.values())
    return entropy

# AES encryption
def aes_encrypt_decrypt(text):
    key = get_random_bytes(16)  # AES-128
    cipher = AES.new(key, AES.MODE_CBC)
    iv = cipher.iv
    ct_bytes = cipher.encrypt(pad(text.encode(), AES.block_size))

    # Decrypt
    cipher = AES.new(key, AES.MODE_CBC, iv=iv)
    decrypted_text = unpad(cipher.decrypt(ct_bytes), AES.block_size).decode()

    return ct_bytes, decrypted_text, key

# 3DES encryption
def triple_des_encrypt_decrypt(text):
    key = DES3.adjust_key_parity(get_random_bytes(24))  # 3DES key
    cipher = DES3.new(key, DES3.MODE_CBC)
    iv = cipher.iv
    ct_bytes = cipher.encrypt(pad(text.encode(), DES3.block_size))

    # Decrypt
    cipher = DES3.new(key, DES3.MODE_CBC, iv=iv)
    decrypted_text = unpad(cipher.decrypt(ct_bytes), DES3.block_size).decode()

    return ct_bytes, decrypted_text, key

# DES encryption
def des_encrypt_decrypt(text):
    key = get_random_bytes(8)  # DES key
    cipher = DES.new(key, DES.MODE_CBC)
```

```
    iv = cipher.iv
    ct_bytes = cipher.encrypt(pad(text.encode(), DES.block_size))

    # Decrypt
    cipher = DES.new(key, DES.MODE_CBC, iv=iv)
    decrypted_text = unpad(cipher.decrypt(ct_bytes), DES.block_size).decode()

    return ct_bytes, decrypted_text, key

# Blowfish encryption
def blowfish_encrypt_decrypt(text):
    key = get_random_bytes(16)  # Blowfish key
    cipher = Blowfish.new(key, Blowfish.MODE_CBC)
    iv = cipher.iv
    ct_bytes = cipher.encrypt(pad(text.encode(), Blowfish.block_size))

    # Decrypt
    cipher = Blowfish.new(key, Blowfish.MODE_CBC, iv=iv)
    decrypted_text = unpad(cipher.decrypt(ct_bytes),
Blowfish.block_size).decode()

    return ct_bytes, decrypted_text, key

# Sample text
sample_text = "I love Cryptography"

# Initialize lists to hold results for plotting
algorithms = ['AES', '3DES', 'DES', 'Blowfish']
keys = []
encrypted_texts = []
decrypted_texts = []
key_entropies = []

# Perform encryption and store results
for algo in algorithms:
    if algo == 'AES':
        encrypted, decrypted, key = aes_encrypt_decrypt(sample_text)
    elif algo == '3DES':
        encrypted, decrypted, key = triple_des_encrypt_decrypt(sample_text)
    elif algo == 'DES':
        encrypted, decrypted, key = des_encrypt_decrypt(sample_text)
    elif algo == 'Blowfish':
        encrypted, decrypted, key = blowfish_encrypt_decrypt(sample_text)

    # Store results
```

```
    keys.append(key)
    encrypted_texts.append(encrypted)
    decrypted_texts.append(decrypted)
    key_entropies.append(calculate_entropy(key))

# Display Results
print("\n=== Encryption Results ===")
for algo, encrypted, decrypted, key, entropy in zip(algorithms, encrypted_texts,
decrypted_texts, keys, key_entropies):
    print(f"\nAlgorithm: {algo}")
    print(f"Encrypted Text: {encrypted}")
    print(f"Decrypted Text: {decrypted}")
    print(f"Key: {key}")
    print(f"Key Entropy: {entropy:.4f}")

# Prepare data for plotting
# Plotting the key entropy values
plt.figure(figsize=(8, 5))
plt.bar(algorithms, key_entropies, color='orange')
plt.xlabel('Encryption Algorithms')
plt.ylabel('Key Entropy')
plt.xticks(rotation=15)

# Display the plot
plt.tight_layout()
# plt.savefig("plot.png")  # Save as PNG
# plt.savefig("plot.jpg", dpi=300)  # Save as JPG with higher resolution
# plt.savefig("plot.pdf")
plt.savefig("plot.eps", format="eps", dpi=300)

plt.show()
```

## Individual Contribution:

**Karan Kumar Chauhan (21BCE10243):**

- Responsible for coding and algorithmic analysis aspects of the research.
- Implemented various cryptographic algorithms and analyzed their efficiency in terms of speed, security, and computational complexity.
- Collaborated in writing and testing code for encryption and decryption processes, ensuring their correctness and robustness.
- Responsible for compiling and interpreting experimental results, presenting them in a structured manner.
- Contributed as the author for conclusion, summarizing key findings and discussing future research directions.

**Abhishek Kumar (21BCE10084)**:

- Responsible for in-depth analysis on the fundamentals of information theory.
- Responsible for gathering information on key concepts such as entropy, mutual information, and channel capacity.
- Contributed in studying various theoretical models that quantify information and its transmission efficiency.
- Collaborated in implementing various cryptographic algorithms.
- Demonstrated how the applications of information theory principles to cryptography through examples.

**Deep Radadiya (21BCE11343):**

- Contributed on exploring and analyzing different cryptographic models.
- Provided an in-depth analysis on symmetric and asymmetric encryption techniques, hash functions, and key exchange mechanisms.
- Researched on classical cryptographic methods such as RSA, AES, and ECC, along with emerging trends in quantum-resistant cryptography.
- Examined the strengths and weaknesses of each model, assessed their applicability in various security scenarios.
- Illustrated real-world implementations of cryptographic models with relevant case studies.

**Vikash Kumar Sinha(21BCE11500):**

- Responsible for analyzing how principles from information theory enhance cryptographic techniques, particularly in the context of secure encoding and transmission.
- Contributed by studying Shannon's theory of perfect secrecy, error correction codes, and the application of entropy in cryptographic key generation and extracting the important information.
- Demonstrated how information-theoretic approaches contribute to the development of robust encryption schemes by linking theoretical frameworks with practical security mechanisms.
- Contributed by providing a deeper understanding of how cryptographic security can be mathematically assured through information theory.
- Contributed by explaining how cryptographic models safeguard information in digital communication systems.

**Amul Gupta(21BCE11606):**

- Responsible for explaining the significance of cryptographic techniques and their interplay with information theory.
- Responsible for literature review and highlighting the existing challenges in cryptographic systems.
- Responsible for research to bridge the gap between theoretical concepts and real-world applications.
- Collaborated in compiling the code and results of various cryptographic algorithms.
- Responsible for providing a concise and comprehensive summary of the methodology and key findings in the research paper.

# CHAPTER 5: CONCLUSION

The secure communication system developed in this project integrates encryption, decryption, and entropy-based security analysis within a Jupyter Notebook environment. By leveraging both symmetric and asymmetric encryption techniques, the system ensures confidentiality and resistance against cryptographic attacks. The analysis of entropy and probability distributions validates the randomness and security strength of the encryption processes.

Through experimental results, we demonstrated that higher entropy values correspond to increased security, reducing the likelihood of successful cryptanalysis. The implementation of Shannon's secure communication model further enhances data protection by simulating adversarial scenarios and testing encryption robustness against brute-force attacks. Additionally, the performance evaluation of different cryptographic algorithms highlights the trade-offs between security and computational efficiency, providing insights into their practical applications.

The findings of this study emphasize the importance of information-theoretic security measures in cryptographic systems. The developed framework serves as a foundation for further research in secure communications, enabling enhancements in encryption methodologies and security assessments. Future work may focus on optimizing algorithm efficiency, integrating machine learning for cryptographic key predictions, and expanding the system to real-world secure messaging applications.

In conclusion, the system successfully demonstrates a robust approach to secure data transmission using cryptographic principles and entropy-based security analysis, reinforcing the necessity of strong encryption mechanisms in modern communication systems.

## References and Publications:


1. C. E. Shannon, “A mathematical theory of communication,” The Bell System Technical Journal, vol. 27, pp. 379–423, 623–656, Jul. 1948.

2. S. Alam, S. Thakor, and S. Abbas, “On enumerating distributions for associated vectors in the entropy space,” in Proceedings of the International Symposium on Information Theory and its Applications (ISITA), IEICE, Singapore, Oct. 2018, pp. 65–69.

3. S. Alam, S. Thakor, and S. Abbas, “Inner bounds for the almost entropic region and network code construction,” IEEE Transactions on Communications, vol. 69, no. 1, pp. 19–30, 2021. DOI: 10.1109/TCOMM.2020.3030055.

4. S. Naeem, S. Anam, A. Ali, M. Zubair, and M. M. Ahmed, A brief history of information theory by Claude Shannon in data communication, 8th ed., Basel, Switzerland: MDPI, Jan. 2023, pp. 1–31.

5. K. Sayood, Information theory and cognition: A review. Morgan Claypool Publishers, Sep. 2018, pp. 2–5.
6. S. Dehaene, M. I. Posner, and D. M. Tucker, “Localization of a neural system for error detection and compensation,” NeuroImage, vol. 5, no. 5, Sep. 1994.

7. A. Rapoport and W. J. Horvath, “The theoretical channel capacity of a single neuron as determined by various coding systems,” Information and Control, vol. 3, no. 4, pp. 335–350, Dec. 1960.

8. A. M. Abdullah, “Advanced encryption standard (AES) algorithm to encrypt and decrypt data,” International Journal of Computer Science and Mobile Computing, Jun. 2017.

9. D. Chowdhury, A. Dey, R. Garai, et al., “Decrypt: A 3DES-inspired optimized cryptographic algorithm,” Journal of Ambient Intelligence and Humanized Computing, vol. 14, no. 5, 2023.

10. K. Meera and N. Selvaganesan, “Study on various encryption/decryption algorithms for secure communication using chaotic based hashed key,” in 2023 Ninth Indian Control Conference (ICC), Visakhapatnam, India: IEEE, 2023, pp. 79–84.

11. A. M. Alabaichi, F. Ahmad, and R. Mahmod, “Security analysis of Blowfish algorithm,” International Journal of Computer Science and Mobile Computing, Sep. 2013.

12. F. J. Aufa, Endroyono, and A. Affandi, “Security system analysis in combination method: RSA encryption and digital signature algorithm,” in 2018 4th International Conference on Science and Technology (ICST), Yogyakarta, Indonesia: IEEE, 2018, pp. 1–5.

13. M. R. Ramadhan, S. Mandala, and F. A. Yulianto, “Analysis and implementation of digital signature algorithm in PDF document,” in 2023 11th International Conference on Information and Communication Technology (ICoICT), Melaka, Malaysia: IEEE, 2023, pp. 11–16.

14. M. Gobi, R. Sridevi, and R. R. Priyadharshini, “A comparative study on the performance and the security of RSA and ECC algorithm,” International Journal of Advanced Science and Technology, pp. 168–171, Oct. 2020.

15. U. Maurer, Information-theoretic cryptography. Springer-Verlag Berlin Heidelberg, 1999, pp. 49–62.

16. C. E. Shannon, “Communication theory of secrecy systems,” Bell System Technical Journal, vol. 28, pp. 656–715, Oct. 1949.

17. U. Maurer, “The strong secret key rate of discrete random triples,” in Lecture Notes in Computer Science, Springer, 1994, pp. 271–285.

18. K. S. McCurley and C. D. Ziegler, Advances in cryptology (Lecture Notes in Computer Science). Springer, 1998, vol. 1440.

19. A. D. Wyner, “The wire-tap channel,” Bell System Technical Journal, Oct. 1975.

20. I. Csiszar and J. Korner, “Broadcast channels with confidential messages,” IEEE Transactions on Information Theory, vol. 24, no. 3, pp. 339–348, May 1978.

21. U. M. Maurer, “Secret key agreement by public discussion from common information,” IEEE Transactions on Information Theory, vol. 39, no. 3, pp. 733–742, May 1993.