Cryptography with Information Theory

Sultan Alam, Vikash Kumar Sinha, Karan Kumar Chauhan
School of Computer Science and Engineering
VIT Bhopal University
Sehore, India
mohammadsultanalam, vikash.kumar2021,@vitbhopal.ac.in

Vikash Kumar Sinha
School of Computer Science and Engineering
VIT Bhopal
Sehore, India
vikash.kumar2021@vitbhopal.ac.in

Karan Kumar Chauhan
School of Computer Science and Engineering
VIT Bhopal
Sehore, India
karan.kumar2021@vitbhopal.ac.in

***Abstract*—This research examines the process of key generation and entropy levels in commonly used symmetric and asymmetric cryptographic algorithms. The symmetric algorithms analyzed include AES (Advanced Encryption Standard), 3DES (Triple Data Encryption Standard), DES (Data Encryption Standard), and Blowfish, while the asymmetric category consists of RSA (Rivest-Shamir-Adleman), DSA (Digital Signature Algorithm), and ECC (Elliptic Curve Cryptography). The strength of a cryptographic system is significantly influenced by the randomness of its key generation process, which directly affects its resilience against brute-force and cryptanalytic attacks. To measure this randomness, the study utilizes Shannon entropy, a statistical metric that quantifies unpredictability. A programmatic approach was implemented to generate, analyze, and visualize entropy levels across different key sizes.**

The findings indicate that among symmetric encryption methods, 3DES demonstrated the highest entropy, followed by Blowfish and AES, while DES exhibited the lowest entropy, reinforcing its inadequacy for modern security applications. In the case of asymmetric encryption, entropy per byte appeared lower due to the mathematical structures involved in key generation. However, these algorithms compensate by employing larger key sizes, ensuring strong security. Notably, ECC emerged as a highly efficient option, providing robust protection with smaller key sizes, making it suitable for environments with limited computational resources. This study underscores the interplay between entropy, key size, and computational performance, highlighting the importance of selecting encryption techniques that align with security and efficiency requirements.

I. Introduction

In today's digital landscape, safeguarding sensitive data and ensuring secure communication are critical challenges. Cryptographic algorithms serve as the backbone of modern security systems, protecting information from unauthorized access and cyber threats. The reliability of these algorithms largely depends on the randomness and unpredictability of the cryptographic keys they generate. If a key follows a predictable pattern, it becomes vulnerable to cryptanalysis, making randomness a key factor in encryption security.

One widely used metric for evaluating randomness is Shannon entropy, which measures the uncertainty within a system [1]. A high entropy value signifies stronger, more unpredictable keys that can resist brute-force and statistical attacks, whereas a low entropy [2], [3] value may indicate potential weaknesses that attackers could exploit. Given the increasing computational power of modern adversaries, ensuring sufficient entropy in cryptographic key generation is essential for maintaining secure encryption standards.

This study provides a comparative assessment of key generation mechanisms and entropy levels for widely used symmetric and asymmetric cryptographic algorithms. The symmetric encryption methods analyzed include AES (Advanced Encryption Standard), 3DES (Triple Data Encryption Standard), DES (Data Encryption Standard), and Blowfish. On the asymmetric side, the study evaluates RSA (Rivest-Shamir-Adleman), DSA (Digital Signature Algorithm), and ECC (Elliptic Curve Cryptography). These algorithms are examined based on their key randomness, entropy per byte, and overall computational efficiency, offering insights into their effectiveness for secure applications.

Symmetric Encryption and Entropy Analysis Symmetric encryption employs a single key for both encryption and decryption. The randomness of this key directly impacts the security of the encryption process. This study assesses the entropy per byte of keys generated by symmetric algorithms to determine their level of unpredictability.

The findings suggest that 3DES demonstrates the highest entropy due to its multi-layered encryption approach. Blowfish and AES also exhibit strong entropy levels, making them reliable choices for modern encryption needs. However, DES, with its shorter key length and lower entropy, is highly susceptible to brute-force attacks, reinforcing why it is no longer considered secure for modern applications.

Asymmetric Encryption and Entropy Considerations Unlike symmetric encryption, asymmetric cryptographic algorithms use public and private key pairs to encrypt and decrypt data. Although their entropy per byte is generally lower due to their structured mathematical properties, they compensate by employing larger key sizes to enhance security. Among these, ECC stands out as a highly efficient algorithm, offering robust security with smaller key lengths, making it ideal for constrained environments such as IoT devices, mobile

applications, and embedded systems.

Objective of the Study This research aims to analyze the trade-offs between entropy, key size, and computational efficiency across different cryptographic techniques. By comparing symmetric and asymmetric encryption methods, this study highlights the importance of selecting encryption algorithms based on their entropy properties and security needs, ensuring reliable protection in diverse applications.

II. Basics of Information Theory

Information theory is all about quantifying and understanding the flow of information. It helps us measure and analyze how much information is being transmitted, stored, or processed. It is a field founded by Claude Shannon in 1948 [1], forms the mathematical foundation for understanding the transmission, storage, and processing of information. It deals with quantifying information, studying the limits of communication systems, and designing methods to efficiently transmit data, especially in the presence of noise.

Think of it as a way to put a number on the amount of uncertainty or surprise in a message or signal. There are many information theoretic quantities that we will be referring to for further understanding during this research.

A. Self Information Theory

Imagine we have a discrete random variable, which we can think of as a box of surprises that can take on different values. Let's say this box can contain one of several items

$$\{x_1, x_2, \ldots, x_N\}$$

Each of these items has a certain probability associated with it, which we denote here as $P_X(x_i)$. This probability tells us how likely it is for the random variable X to reveal the item x_i when we open the box [4].

Now, we will see how Self-Information helps here. Self-information, denoted a $I(x_i)$, measures how much information we gain when we learn that the random variable X has indeed taken on the value x_i. In simpler terms, it tells us how surprising or informative it is to find out that the box contains x_i[5].

The formula we get for calculating self-information is:

$$I(x_i) = -\log(P_X(x_i))$$

Now, if the events represented by X and Y are independent, the joint probability of both events occurring together can be calculated as the product of their individual probabilities:

$$P_(X, Y)(x_i, y_j) = P_X(x_i).P_Y(y_j)$$

Using this relationship, we can rewrite the self-information for the pair:[5].

$$I(x_i, y_j) = -log(P_X(x_i).P_Y(y_j))$$

We can say that the Self-Information associated with independent events is additive [5].

B. Entropy

A fundamental concept in information theory introduced by Claude Shannon, measures the uncertainty or randomness in a system or message. It quantifies how unpredictable or surprising a message is—higher unpredictability means higher entropy. This randomness refers to the lack of pattern or predictability in a sequence of events or data, where no systematic rule governs outcomes. For instance, a fair coin flip with unknown results has high entropy, as the outcome is uncertain, whereas knowing with certainty that a coin will land on heads results in zero entropy. Shannon also extended these ideas to cryptography, introducing the concept of perfect secrecy, where a cipher-text reveals no information about the plain-text, preserving complete uncertainty and randomness. Both concepts highlight how unpredictability plays a crucial role in information theory and secure communication [1][5].

C. Secrecy

Secrecy is defined as protection of information from unauthorized access, ensuring that only intended recipients can interpret or understand the data. Understanding entropy helps in analyzing how secure an encryption scheme is. The entropy, denoted as H(X), can be mathematically expressed as:

$$H(X) = E[I(x_i)] = \sum I(x_i)P_X(x_i)$$

Substituting the definition of self-information, we have:

$$H(X) = \sum_i P_X(x_i) \log\left(\frac{1}{P_X(x_i)}\right)$$

$$= -\sum_i P_X(x_i) \log P_X(x_i)$$

D. Mutual Information

Another important concept is mutual information. This measures how much information two variables or messages share with each other. It's like a way to quantify how much one thing tells you about another.it can be used to assess how much information an attacker can gain about the plain-text from the cipher-text. For example, if you know the weather forecast, that gives you some information about whether you should bring an umbrella[5].

The mutual information between the weather forecast and your umbrella decision is a measure of how much the forecast helps you predict the umbrella situation.

Let's consider a practical application of average mutual information in the context of communication systems. If we think of X as the input to a noisy communication channel and Y as the output from that channel, we can define the capacity of the channel. This capacity represents the maximum amount of information that can be reliably transmitted through the channel.

To find this capacity, we maximize the average mutual information $I(X;Y)$ over all possible distributions of the input X [5]:

$$C = \max_{P(X)} I(X;Y)$$

C: This is the channel capacity, which is the maximum rate at which information can be transmitted over a communication channel without error.

$\max_{P(X)}$:This indicates that the mutual information $I(X;Y)$ is maximized over all possible probability distributions $P(X)$ of the input random variable X. In other words, you are trying to find the input distribution $P(X)$ that maximizes the mutual information between X and Y.

We also have the idea of relative entropy, also known as Kullback-Leibler divergence. This is a way to measure how different two probability distributions are from each other. It's like a distance between two ways of describing the same thing. Imagine you have two different weather forecasts - one from the TV and one from the radio. The relative entropy between these two forecasts tells you how much they disagree with each other on average.

Shannon-Hartley theorem derived an equation which gives the idea of relative entropy[1].

$$C = B \log_2 (1 + \text{SNR})$$

Here, the Variables means:
C: Channel capacity (measured in bits per second, bps)
B: Bandwidth of the channel (measured in Hertz, Hz)
SNR: Signal-to-noise ratio (dimensionless)

E. Error correction and Detection

Error detection involves identifying the presence of errors in the transmitted data. One common method that is being used from a long time is Cyclic Redundancy Check (CRC), which generates a checksum from the data and appends it to the message[1], [5].

The receiving system performs the same calculation and compares it to the checksum. If they differ, an error is detected.

Given a data sequence represented as a polynomial $D(x)$ and a generator polynomial $G(x)$, the CRC is calculated as follows:

$$R(x) = D(x) \times x^n \mod G(x)$$

Here, n is the degree of the generator polynomial $G(x)$, and $R(x)$ is the remainder, which serves as the CRC code.

The transmitted message is then:

$$T(x) = D(x) \cdot x^n + R(x)$$

If the receiver receives the remainder of $T(x)$ divided by $G(x)$ as zero, the data is considered error-free.

In case of error correction, it not only detects errors but also corrects them, ensuring that the received data matches the original transmission. Error-Correcting Codes (ECC) like Hamming Codes and Reed-Solomon Codes are often used in Information Theory[6].

For a message with m data bits and r parity bits, the relationship between them is:

$$2^r \geq m + r + 1$$

F. Channel Capacity

As defined by the Shannon-Hartley theorem, the concept refers to the maximum rate at which information can be reliably transmitted over a communication channel. This is the maximum rate at which information can be transmitted reliably through a communication channel, without errors. It's like the speed limit on an information highway. If you try to send information too fast, it gets garbled, but if you go slower, you can transmit it more accurately. The channel capacity is the sweet spot in the middle[1], [5].

The fundamental theorem of information theory states that if we send messages at a rate that is lower than the channel's maximum capacity, we can use error correction codes to make the chances of mistakes extremely small.

In other words, if we want to send messages without errors, we need to make sure we're not sending too much information at once.

To maximize the efficiency of our communication, we should make sure that the symbols we send are chosen in a way that maximizes the amount of information they convey. This means that each symbol should be used with a probability that is proportional to its importance.. By doing this, we can ensure that the average amount of information transmitted is as high as possible.

An isolated channel input symbol a_i should occur with a probability p_i such that the average mutual information is maximized.

For the noisy channel, with transition matrix Q, the channel capacity, C, is expressed mathematically by

$$C = \max_P \sum_{i=0}^{i-1} \sum_{j=0}^{j-1} P_i Q_{j|i} \log \frac{Q_{j|i}}{\sum_i P_i Q_{j|i}}$$

where:

P: Represents the probability distribution over the input symbols. The notation max_P indicates that we are looking for the input probability distribution that maximizes the sum.

P_i: The probability of sending the i^th input symbol from the input alphabet. It is part of the probability distribution over the input symbols that we are optimizing.

$Q_(j|i)$: The conditional probability that the channel outputs symbol j when input symbol i is transmitted. This value is derived from the transition matrix Q.

$\sum_i P_i Q_{j|i}$: This sum represents the probability of receiving output symbol j considering all possible input symbols and their probabilities. It is the marginal probability of output symbol j.

where the maximum is over all probability distributions on the input alphabet. For simple channels, the capacity can be evaluated by finding the maximum analytically.

However, for more complex channels, there are advanced computational methods to find the maximum efficiently.

The capacity of the binary symmetric channel with transition matrix: [1], [5], [7].

$$\begin{bmatrix} 1 & -p & p \\ p & 1 & -p \end{bmatrix}$$

is

$$C = 1 + P \log_2 P + (1 + P) \log_2 (1 - P)$$

bits per input symbol. The capacity of the binary erasure channel with erasure probability q is

$$C = 1 - q$$

III. Cryptography models

Cryptographic models are frameworks and methodologies used to design, analyze, and implement secure communication systems. These models are critical in ensuring confidentiality, integrity, authenticity, and non-repudiation of data in the digital world. The key cryptographic models are:

- Asymmetric Key Cryptography
- Symmetric Key Cryptography

A. Symmetric Encryption

Symmetric encryption is a method where the same key is used for both encryption and decryption. This means that the sender and the receiver must both have access to the same secret key to encrypt and decrypt the data. The process is relatively fast and straightforward, making it suitable for encrypting large amounts of data.

Key Characteristics

- **Single Key:** The same key is used to encrypt and decrypt the data, which means secure key distribution is critical.
- **Speed:** Symmetric encryption is generally faster than asymmetric encryption, making it ideal for large-scale data encryption.

Common symmetric encryption algorithms include AES (Advanced Encryption Standard), DES (Data Encryption Standard), 3DES (Triple Data Encryption Standard), and Blowfish.

Pros

- **Efficiency:** Symmetric encryption is computationally efficient, especially for encrypting large datasets or streams of data.
- **Simplicity:** Implementing symmetric encryption is generally simpler and requires less computational power compared to asymmetric encryption.

Cons

- **Key Distribution Problem:** The primary challenge with symmetric encryption is securely distributing the key between parties. If the key is intercepted, the security of the encrypted data is compromised.
- **Scalability:** In environments where many parties need to communicate securely, key management becomes complex because a unique key is required for each pair of users.

Use Cases

- **File Encryption:** Symmetric encryption is often used to encrypt files stored on a disk.
- **VPNs and Encrypted Tunnels:** Virtual Private Networks (VPNs) frequently use symmetric encryption to create secure tunnels for data transmission.
- **TLS/SSL:** In the initial stages of secure communication, symmetric encryption is often used once a secure session is established.

1) AES (Advanced Encryption Standard): The Advanced Encryption Standard (AES) is a symmetric key encryption algorithm established by the U.S. National Institute of Standards and Technology (NIST) in 2001. It is widely used for securing sensitive data. AES operates on fixed block sizes of 128 bits and supports key sizes of 128, 192, or 256 bits. The encryption process involves multiple rounds of transformation based on the key, making it resistant to various forms of cryptographic attacks[8].

Key Features

- **Symmetric Key Encryption:** The same key is used for both encryption and decryption.
- **Block Cipher:** AES encrypts data in fixed-size blocks of 128 bits.
- **Variable Key Lengths:** Supports key lengths of 128, 192, and 256 bits.
- **Substitution-Permutation Network (SPN):** AES employs a combination of substitution (S-box) and permutation operations to achieve diffusion and confusion.

Algorithm

Key Expansion: The original encryption key is expanded into multiple round keys. The number of rounds depends on the key size:

- 10 rounds for 128-bit keys
- 12 rounds for 192-bit keys
- 14 rounds for 256-bit keys

Initial Round: **AddRoundKey:** Each byte of the block is combined with the corresponding byte of the round key using the XOR operation.

Main Rounds: Each main round consists of four steps:

- **SubBytes:** Non-linear substitution step where each byte is replaced with a corresponding byte from the S-box.
- **ShiftRows:** Row-wise permutation where the bytes in each row are shifted to the left.
- **MixColumns:** Mixing operation that combines the bytes of each column using polynomial arithmetic.
- **AddRoundKey:** The round key is again combined with the data block using XOR.

Final Round: The final round consists of only three steps:

- SubBytes
- ShiftRows
- AddRoundKey

2) 3DES (Triple DES): 3DES, or Triple Data Encryption Standard, is a symmetric-key block cipher used in cryptography to provide secure data encryption. It is an enhancement of the original DES (Data Encryption Standard) algorithm. 3DES applies the DES algorithm three times to each data block, hence the name "Triple DES". The goal is to increase the security of DES, which was considered vulnerable due to its relatively short key length[9].

3DES works with 64-bit block sizes and uses three 56-bit keys, resulting in a total effective key length of 168 bits. The three keys are often denoted as K1, K2, and K3. Depending on the key usage, there are two main modes of 3DES:

- **3-key 3DES:** All three keys are distinct ($K_1 \neq K_2 \neq K_3$).
- **2-key 3DES:** K_1 and K_3 are the same, reducing the key length to 112 bits.

Key Features

- **Improved Security:** 3DES provides significantly better security compared to DES by using multiple encryption steps.
- **Backward Compatibility:** Since 3DES is based on DES, systems that were using DES could easily transition to 3DES.
- **Slower Performance:** 3DES is slower than DES due to the multiple encryption and decryption steps, but it is still widely used in legacy systems.

Algorithm

Let:

- $E()$ be the encryption function using DES.
- $D()$ be the decryption function using DES.
- P be the plaintext (64-bit block).
- C be the ciphertext (64-bit block).
- K_1, K_2, K_3 be the three different keys (each 56 bits).

Encryption Phase: The encryption process for 3DES is as follows:

- Step 1: Encrypt the plaintext P using key K_1.
- Step 2: Decrypt the result of step 1 using key K_2.
- Step 3: Encrypt the result of step 2 using key K_3.
- The final output C is the ciphertext.

Decryption Phase: To decrypt the ciphertext, the process is reversed:

- Step 1: Decrypt the ciphertext C using key K_3.
- Step 2: Encrypt the result of step 1 using key K_2.
- Step 3: Decrypt the result of step 2 using key K_1.
- The final output P is the original plaintext.

3) DES (Data Encryption Standard) : The Data Encryption Standard (DES) is a symmetric key block cipher that is one of the most widely used encryption algorithms. Developed by IBM in the 1970s and later adopted as a federal standard by the National Institute of Standards and Technology (NIST) in 1977, DES operates on 64-bit blocks of plaintext and uses a 56-bit key (after dropping 8 parity bits from the original 64-bit key) to encrypt and decrypt data.

DES employs a series of complex transformations, including permutations and substitutions, over multiple rounds (16 rounds in total) to achieve security. Each round uses a different subkey derived from the main key. DES is now considered insecure for many applications due to its relatively short key length, which makes it vulnerable to brute-force attacks[10].

Main Components of DES

- **Initial Permutation (IP):** The 64-bit plaintext is permuted using a predefined table.
- **Rounds (16 rounds):** The permuted plaintext is divided into two 32-bit halves, which undergo 16 identical rounds of processing.
- **Feistel Structure:** The round function in DES uses the Feistel network, where only half of the data block (32-bit) is transformed using the other half. This structure ensures invertibility, making decryption possible.
- **Expansion, Substitution, and Permutation:**
  - The 32-bit right half is expanded to 48 bits.
  - The expanded half is XORed with a 48-bit round key.
  - The result passes through substitution boxes (S-boxes), reducing it back to 32 bits.
  - A permutation is applied.
- **Key Schedule:** The 56-bit key is divided, permuted, and shifted to produce 16 round keys.
- **Final Permutation (FP):** After 16 rounds, the final permutation (the inverse of the initial permutation) is applied, resulting in the ciphertext.

Algorithm

Key Generation:

- The 64-bit key is reduced to 56 bits by dropping the parity bits using the PC-1 (Permuted Choice 1) table.
- The 56-bit key is split into two 28-bit halves.
- Each half is shifted left by 1 or 2 bits depending on the round number (using a predefined schedule).
- After shifting, the halves are combined, and PC-2 (Permuted Choice 2) is applied to produce the 48-bit round key.
- This process is repeated 16 times to generate 16 round keys.

Encryption:

- The 64-bit plaintext is permuted using the Initial Permutation (IP) table.
- The permuted block is split into two 32-bit halves (left half and right half).
- For each round i from 1 to 16:
  - Apply the Feistel Function to the right half, using the round key.
  - **Expansion:** The 32-bit right half is expanded to 48 bits.
  - **Key Mixing:** The expanded right half is XORed with the 48-bit round key.

  - **Substitution:** The XOR result is passed through eight substitution boxes (S-boxes), reducing it back to 32 bits.
  - **Permutation:** A fixed permutation is applied to the 32-bit output.
  - The result of the Feistel function is XORed with the left half, and the new right half becomes the old left half.
- After 16 rounds, the left and right halves are swapped and combined.
- The combined 64-bit block undergoes the Final Permutation (FP), producing the ciphertext.

Decryption: Decryption is identical to encryption, but the round keys are applied in reverse order. Since DES uses the Feistel structure, the decryption process is inherently reversible.

4) Blowfish: Blowfish is a symmetric-key block cipher designed by Bruce Schneier in 1993, primarily for encryption purposes. It is known for being fast, compact, and simple, making it a popular choice for embedded systems and applications with limited memory. Blowfish operates on 64-bit blocks and allows a variable-length key ranging from 32 bits to 448 bits[11].

Key Features

- **Symmetric Key Cipher:** Same key is used for both encryption and decryption.
- **Block Size:** 64-bit block cipher, meaning it encrypts data in chunks of 64 bits at a time.
- **Key Size:** The key can vary from 32 bits to 448 bits, offering flexibility in security levels.
- **Feistel Network Structure:** Blowfish uses 16 rounds of Feistel structure, ensuring diffusion of bits from both input and key.
- **Subkeys:** It employs 18 subkeys derived from the input key, used during the encryption and decryption process.
- **S-Boxes:** It uses four fixed, large S-Boxes, which introduce confusion by mapping 8-bit input values to 32-bit output values.

Algorithm

Key Expansion: The key expansion step involves generating subkeys (P-array and S-boxes) from the initial key. Blowfish uses a total of:

- 18 32-bit subkeys stored in a P-array: P1, P2, ..., P18
- Four 32-bit S-boxes each containing 256 entries (1024 32-bit entries total).

Key Expansion Algorithm:

1) Initialize the P-array and S-boxes with predefined hexadecimal values (constants derived from the hexadecimal digits of pi).
2) XOR the initial key with the P-array. If the key length is shorter than the required length, repeat the key until it matches the length of the P-array.
3) Encrypt the all-zero string (0x0000000000000000) using the Blowfish algorithm.
4) Use the resulting ciphertext to replace P1 and P2.
5) Encrypt the updated ciphertext and use the result to update P3 and P4.
6) Repeat the process until the entire P-array and S-boxes are filled with the results of successive encryptions.

Encryption: Blowfish encryption uses a 16-round Feistel structure. The encryption function F is applied in each round, and the subkeys are XORed with the input.

Blowfish Encryption Algorithm:

1) Split the 64-bit plaintext block into two 32-bit halves: L (left) and R (right).
2) For rounds 1 to 16:
   - $L = L \oplus P_i$
   - $R = R \oplus F(L)$ (where F is a function that uses the S-boxes to transform L)
   - Swap L and R
3) After the 16 rounds, swap back L and R to their original positions.
4) Perform the final transformation:
   - $R = R \oplus P_{17}$
   - $L = L \oplus P_{18}$
5) Concatenate L and R to form the 64-bit ciphertext block.

F Function (used in each round):

1) The 32-bit input is divided into four 8-bit quarters.
2) Each 8-bit value is used as an index into the four different S-boxes.
3) The results from the S-boxes are added, XORed, and combined to produce the 32-bit output.

Decryption: Blowfish decryption is almost identical to the encryption process, except that the P-array subkeys are applied in reverse order. This ensures efficient symmetric-key operation where the same algorithm can be used for both encryption and decryption.

Algorithm	Developed By	Year Developed	Key Size	Block Size
DES	IBM	1975	56 bits	64 bits
3DES	IBM	1998	112 or 168 bits	64 bits
AES	NIST	2001	128, 192, or 256 bits	128 bits
Blowfish	Bruce Schneier	1993	32 to 448 bits	64 bits

TABLE I
COMPARISON TABLE FOR DIFFERENT SYMMETRIC KEY ALGORITHMS

B. Asymmetric Encryption

Asymmetric encryption, also known as public-key cryptography, uses two different keys: one for encryption (public key) and another for decryption (private key). The public key is shared openly, while the private key is kept secret by the owner. This method allows secure communication even if the public key is shared openly because only the corresponding private key can decrypt the message.

Key Characteristics

- **Two Keys:** A public key encrypts the data, and a private key decrypts it. The private key is never shared.
- **Security:** Asymmetric encryption provides a solution to the key distribution problem seen in symmetric encryption since no secret keys need to be transmitted or shared.
- **Examples:** Well-known asymmetric encryption algorithms include RSA (Rivest-Shamir-Adleman), ECC (Elliptic Curve Cryptography), and DSA (Digital Signature Algorithm).

Pros

- **Key Distribution:** Asymmetric encryption solves the problem of secure key exchange since only the public key needs to be shared, while the private key remains secure.
- **Authentication:** It provides not only confidentiality but also authenticity and integrity, ensuring that the sender of a message is who they claim to be through digital signatures.

Cons

- **Slower:** Asymmetric encryption is computationally more expensive than symmetric encryption, making it less suitable for encrypting large amounts of data.
- **Complexity:** The cryptographic operations involved in asymmetric encryption are more complex and require more computational resources.

Use Cases

- **Digital Signatures:** Asymmetric encryption is commonly used to create digital signatures, verifying the identity of the sender and ensuring message integrity.
- **Key Exchange Protocols:** Asymmetric encryption is often used in protocols like TLS/SSL to securely exchange symmetric encryption keys.
- **Email Encryption:** Tools like PGP (Pretty Good Privacy) use asymmetric encryption to secure email communications.

1) RSA (Rivest-Shamir-Adleman): RSA is a public-key cryptosystem used for secure data transmission. It is one of the first practical cryptosystems and is widely used for secure communication, especially for encrypting sensitive data and authenticating digital signatures.

The security of RSA comes from the difficulty of factoring large composite numbers, specifically the product of two large prime numbers. The algorithm is based on the mathematical properties of modular exponentiation and number theory, particularly Euler's Totient function and Fermat's Little Theorem [12].

In RSA, each user generates two keys:

- **Public key:** Used for encrypting messages and can be distributed to anyone.
- **Private key:** Used for decrypting messages and must be kept secret.

The RSA model has two main processes: key generation and encryption/decryption.

Algorithm

Key Generation:

1) Select two distinct large prime numbers, p and q.
2) Compute n, the modulus for both the public and private keys:
$$n = p \times q$$
3) Compute Euler's Totient function $\varphi(n)$, where:
$$\varphi(n) = (p-1) \times (q-1)$$
4) Choose an integer e such that $1 < e < \varphi(n)$ and $\gcd(e, \varphi(n)) = 1$, meaning e and $\varphi(n)$ are coprime. Typically, e is chosen as 65537 for efficiency.
5) Compute the private key exponent d, which is the modular multiplicative inverse of e modulo $\varphi(n)$:
$$d \times e \equiv 1 \pmod{\varphi(n)}$$
In other words, d satisfies:
$$d = e^{-1} \pmod{\varphi(n)}$$

Public Key: The public key is (n, e).

Private Key: The private key is (n, d).

Encryption: **Input:** Message M, which is an integer such that $0 \leq M < n$.

Process: The ciphertext C is computed as:
$$C = M^e \pmod{n}$$

Output: The encrypted message C.

Decryption: **Input:** Ciphertext C.

Process: The original message M is recovered using the private key d:
$$M = C^d \pmod{n}$$

Output: The decrypted message M.

2) DSA: Digital Signature Algorithm (DSA) is a widely used public key cryptosystem in cryptography. It provides data integrity, authentication, and non-repudiation through digital signatures. DSA is based on mathematical problems such as modular exponentiation and discrete logarithms, making it computationally infeasible to forge a valid signature without the private key.

In Cryptography, DSA is used to generate a digital signature by encrypting a message hash with a private key, which can be verified using the corresponding public key. The integrity of the message is ensured because the signature is tied to both the message and the key.

In Information Theory, DSA ensures the security of transmitted data by preventing unauthorized modifications and providing a verification mechanism that confirms the authenticity and origin of the message [13].

Algorithm

Key Generation: **Select Parameters:**

- Select a large prime p.
- Select another prime q such that q divides $p-1$.
- Select a number g such that $g = h^{(p-1)/q} \mod p$, where h is any number between 1 and $p-1$.

Private Key Generation:

- Choose a random private key x, where $1 < x < q$.

Public Key Generation:

- Calculate the public key y, where $y = g^x \mod p$.
- The public key is (p, q, g, y) and the private key is x.

Signature Generation

Message Hashing:

- Hash the message M to generate a message digest $H(M)$, typically using SHA-1 or other hash algorithms.

Generate Random k:

- Select a random number k such that $1 < k < q$.

Calculate Signature Components:

- $r = (g^k \mod p) \mod q$.
- $s = \left(k^{-1} \cdot (H(M) + x \cdot r)\right) \mod q$.

Output the Signature:

- The signature is the pair (r, s).

Signature Verification

Verify Signature Components:

- Given the public key (p, q, g, y) and the signature (r, s), compute the following:
- Verify that $0 < r < q$ and $0 < s < q$.
- Compute $w = s^{-1} \mod q$.
- Calculate $u_1 = (H(M) \cdot w) \mod q$ and $u_2 = (r \cdot w) \mod q$.

Verification Condition:

- Compute $v = (g^{u_1} \cdot y^{u_2}) \mod p \mod q$.
- If $v = r$, the signature is valid. Otherwise, it is invalid.

3) Elliptic Curve Cryptography (ECC): Elliptic Curve Cryptography (ECC) is a modern public-key cryptosystem that offers strong security with smaller key sizes compared to other methods such as RSA. ECC is based on the algebraic structure of elliptic curves over finite fields. The underlying mathematical problem that ECC leverages is the difficulty of the Elliptic Curve Discrete Logarithm Problem (ECDLP), which is computationally infeasible to solve with current technologies for large key sizes.

In ECC, security is achieved by generating keys from points on an elliptic curve. Given two points on the curve, their sum is also a point on the curve, and scalar multiplication of a point (adding a point to itself multiple times) forms the basis for cryptographic operations. ECC provides efficient encryption, decryption, key exchange, and digital signature processes, making it highly suitable for systems where computational power, memory, and bandwidth are limited, such as mobile devices and IoT [14].

Algorithm

Key Generation: The input for key generation requires elliptic curve E, base point G on the curve, and a large random integer d (private key). The output will be public key P.

Steps for key generation:

- Select an elliptic curve E defined by an equation $y^2 = x^3 + ax + b$ over a finite field F_q, where a and b are curve constants and q is the field order.
- Choose a base point G on the curve E, which is a known point.
- Choose a large random integer d as the private key.
- Compute the public key $P = dG$ (scalar multiplication of the base point by the private key).

Encryption

The input for encryption requires message M, receiver's public key P, elliptic curve E, and base point G, and the output will be encrypted ciphertext (C_1, C_2).

Steps for encryption:

- Represent the message M as a point on the elliptic curve E (message encoding process).
- Choose a random integer k.
- Compute $C_1 = kG$, which is a point on the curve.
- Compute $C_2 = M + kP$, where P is the receiver's public key.
- The ciphertext is the pair (C_1, C_2).

Decryption

The input for decryption requires ciphertext (C_1, C_2), receiver's private key d, and the output will be the decrypted message M.

Steps for decryption:

- Compute $dC_1 = d(kG) = kP$, which is a point on the elliptic curve.
- Subtract kP from C_2 to recover the message: $M = C_2 - kP$.

Algorithm	Developed By	Year Developed	Key Size	Block Size
DES	IBM	1975	56 bits	64 bits
3DES	IBM	1998	112 or 168 bits	64 bits
AES	NIST	2001	128, 192, or 256 bits	128 bits
Blowfish	Bruce Schneier	1993	32 to 448 bits	64 bits

TABLE II
COMPARISON TABLE FOR DIFFERENT ASYMMETRIC KEY ALGORITHMS

IV. Cryptography with Information Theory

In this section, we explore how cryptography can be enhanced through the principles of information theory. Cryptography has traditionally relied on computational assumptions, such as the difficulty of solving complex mathematical

problems, to ensure security. However, information theory provides an alternative approach to achieving security that does not depend on computational limitations. This section covers key concepts where information theory directly impacts cryptographic systems, ensuring their security and efficiency.

We discuss several applications of information theory in cryptography, encompassing both unconditional and computational security. Unconditionally secure secrecy is reviewed, with a focus on how unconditional security can be practically achieved by exploiting the physical world's inherent uncertainties—such as noise—which prevent any party from having complete information about a system's state. We introduce the concept of an information-theoretic cryptographic, covering a range of primitives like oblivious transfer, noisy channels, and secure multiparty computation. Many findings in information-theoretic cryptography can be viewed as reductions among these primitives [15].

Information theory has played a significant role in cryptography, mainly in establishing pessimistic results, such as determining lower bounds on the size of the secret key required to ensure a certain degree of security. We explore the three key areas where such bounds have been identified: secrecy, secret sharing, and wiretap/broadcast channeling. Consider the model of a symmetric cryptosystem depicted in Fig. 1, which is a generalization of Shannon's model. It includes a secret randomizer, S, known only to the sender of the message X, as well as a public randomizer, R, accessible to everyone, including any potential eavesdroppers.

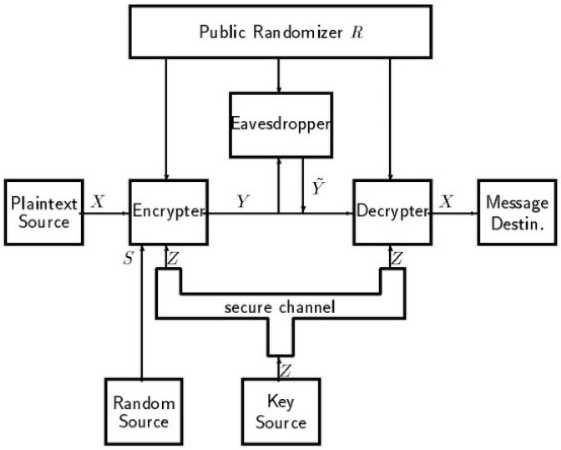

Fig. 1. Model of symmetric cipher with two types of randomizers.

In secure communication, two fundamental and complementary objectives must be met: confidentiality and authenticity. Imagine you're sending a secret message. Confidentiality ensures that a curious eavesdropper cannot learn anything meaningful about the message, keeping your words hidden. On the other hand, authenticity guarantees that even if someone tries to tamper with or forge a message—creating a fake one, like a fabricated Y it won't be accepted as genuine by the recipient. Both goals work hand-in-hand, ensuring that your message stays both private and trustworthy.

A. Information-Theoretic Security in Cryptography

In our exploration of cryptography, we recognize the significance of information-theoretic security as a fundamental shift from traditional approaches that rely on computational hardness. Unlike computational security, which depends on the assumption that certain problems are difficult to solve within practical time limits, information-theoretic security assures us that a cryptographic system remains secure even against an adversary with unlimited computational resources. This form of security is deeply rooted in the principles of information theory, a field pioneered by Claude Shannon, whose work laid the groundwork for many of the concepts we now explore.

1) Perfect Secrecy: A ciphertext is considered perfect if, and only if, the plaintext X and the ciphertext Y, together with the public randomizer R, are statistically independent. In other words, this condition holds if and only if

$$I(X;YR) = 0$$

i.e., the mutual information between X and the joint variables Y and R. The following theorem is an extension of Shannon's theorem [16].

Theorem 1. A cipher can be perfect only if

$$H(Z) \geq H(X)$$

where $H(X)$ denotes the entropy of the random variable X.

Proof. Every cipher must be uniquely decipherable. Therefore,

$$H(X|YZR) = 0$$

The definition of perfect secrecy, $I(X;YR) = 0$, can be stated as,

$$H(X|YR) = H(X)$$

Using the basic expansion rule for uncertainties, and the fact that the removal of given knowledge can only increase uncertainty, we obtain

$$\begin{aligned} H(X) = H(X|YR) &\leq H(XZ|YR) \\ &= H(Z|YR) + H(X|YZR) \\ &= H(Z|YR) \\ &\leq H(Z) \end{aligned}$$

It is important to note that the condition in Theorem 1 does not necessarily ensure a perfect cipher. As will be shown in Section 3.1.3, achieving near-perfect secrecy is feasible even when H(Z)H(X), provided that the model in Figure 1 is slightly adjusted. For example, if the public randomizer is sufficiently large, it becomes impractical (though theoretically possible) for an eavesdropper to intercept the entire string R, resulting in incomplete information, which contradicts the Shannon assumption inherent in the model. Another approach might be to introduce errors in accessing the randomizer; for instance, if R is transmitted via a satellite with a weak signal,

the sender, receiver, and eavesdropper could all experience non-zero bit error probabilities when receiving the data. Remarkably, perfect secrecy can be attained in such realistic conditions even without the use of a secret key Z, and even when the eavesdropper has access to the random bits of R far more reliably than the legitimate parties [17].

2) secret Sharing: Information theory has been recently utilized to establish lower bounds on the share sizes in perfect secret sharing schemes [18]. A perfect secret sharing scheme enables a secret S to be distributed among N participants such that only qualified subsets of participants can reconstruct the secret, while any unqualified subset remains entirely uninformed about S.

Let A represent a set of participants, and an access structure $\mathcal{A}$ on P is defined as a family of subsets of P, such that $\mathcal{A} \subseteq 2^P$. We focus on monotonic access structures, meaning that if $T \in \mathcal{A}$ and $T \subseteq T'$, then $T' \in \mathcal{A}$ as well. For any given access structure, we consider its monotone closure. The share assigned to a participant $P \in P$ is also denoted by P, assuming this notation will not cause confusion. Similarly, the set of shares allocated to a subset $T \subseteq P$ is denoted by T. The conditions for a perfect secret sharing scheme are as follows:

$$T \in \mathcal{A} \implies H(S|T) = 0$$

and

$$T \notin \mathcal{A} \implies H(S|T) = H(S)$$

Consider now the set $P = \{A, B, C, D\}$ of four participants and the access structure consisting of the monotone closure of $\mathcal{A} = \{\{A, B\}, \{B, C\}, \{C, D\}\}$, which is:

$$\mathcal{A} = \{\{A, B\}, \{B, C\}, \{C, D\}, \{A, B, C\}, \{A, B, D\},$$
$$\{A, C, D\}, \{B, C, D\}, \{A, B, C, D\}\}$$

Hence, we have:

$$H(S \mid A, B) = H(S \mid B, C) = H(S \mid C, D) = 0$$

But:

$$H(S \mid A, C) = H(S \mid A, D) = H(S \mid B, D) = H(S)$$

Using these facts, one can derive the lower bound: $H(B, C) \geq 3H(S)$ [18]. This indicates that at least one of the shares held by B or C must be at least 1.5 times the length of S, meaning: $H(B) \geq 1.5H(S)$ or $H(C) \geq 1.5H(S)$ or both. Rather than reiterating the argument from **ref3**, we will present a general proof technique.

Consider the set $P' = P \cup \{S\}$ and the collection of all subsets of P', which forms a partially ordered set. A labeled directed graph can be naturally associated with P' as follows:

- The vertices represent the subsets of P', including the empty set $\emptyset$.
- Two vertices T and T' are connected by a directed edge $T \to T'$ if and only if $T' \neq T$ and $T = T' \cup \{P\}$ for some $P \in P'$.
- Each edge $T \cup \{P\} \to T$ can be labeled with $H(T \cup \{P\} \mid T) = H(P \mid T)$.

For a given access structure $\mathcal{A}$, this labeling implies that an edge $T \cup \{S\} \to T$ is labeled $H(S)$ if and only if $T \notin \mathcal{A}$, and labeled 0 if and only if $T \in \mathcal{A}$.

A proof for establishing a lower bound on the size of the shares in a secret sharing scheme is often framed as a lower bound on $H(T)$ for some $T \subset P$. This lower bound can be obtained by identifying an appropriate path from T to $\emptyset$ in the previously described graph. Along this path:

- The entropies assigned to the edges are added when traversing in the labeled direction.
- The entropies are subtracted when moving in the reverse direction.

The key to deriving such a proof lies in selecting a path that maximizes the traversal of $H(S)$-labeled edges in the positive direction and 0-labeled edges in the negative direction.

Consider the example discussed above. Edges labeled $H(S)$ are:

$$AS \to A, \quad BS \to B, \quad CS \to C, \quad DS \to D,$$
$$ADS \to AD, \quad ACS \to AC, \quad \text{and} \quad BDS \to BD,$$

and the edges labeled 0 are:

$$ABS \to AB, \quad BCS \to BC, \quad CDS \to CD,$$
$$ABCS \to ABC, \quad ABDS \to ABD, \quad ACDS \to ACD,$$
$$BCDS \to BCD, \quad \text{and} \quad ABCDS \to ABCD.$$

Consider now the path:

$$BC \to BCS \to ABCS \to ACS \to AC \to ACD$$
$$\to ACDS \to ADS \to AD \to A \to \emptyset,$$

which can be interpreted as:

$$H(BC) = -H(S \mid BC) - H(A \mid BCS) + H(B \mid ACS)$$
$$+H(S \mid AC) - H(D \mid AC) - H(S \mid ACD) + H(C \mid ADS)$$
$$+H(S \mid AD) + H(D \mid A) + H(A). \quad (1)$$

A closed loop in the graph corresponds to a total value of 0. Therefore, the loop:

$$B \to AB \to ABS \to BS \to B$$

corresponds to the equation:

$$-H(A \mid B) - H(S \mid AB) + H(A \mid BS) + H(S \mid B) = 0. \quad (2)$$

Adding the left side of (2) to the right side of (1), and using:

$$H(S \mid BC) = H(S \mid AB) = H(S \mid ACD) = 0,$$
$$H(S \mid AC) = H(S \mid AD) = H(S \mid B) = H(S),$$
$$H(A) - H(A \mid B) \geq 0, \quad H(D \mid A) - H(D \mid AC) \geq 0,$$
$$H(A \mid BS) - H(A \mid BCS) \geq 0, \quad H(B \mid ACS) \geq 0, \quad \text{and} \quad H(C \mid AD$$

we obtain the desired result:

$$H(BC) \geq 3H(S).$$

This proof technique can be automated and implemented as a graph-theoretic algorithm. Recently, it has been used to identify the first access structure for which, in any secret sharing scheme, at least one share is required to be at least twice the length of the secret [18].

3) Wire-tap and broadcast channels: The first model of this kind was introduced by Wyner in 1975, known as the wire-tap channel [19]. Wyner analyzed a scenario where an eavesdropper, Eve, intercepts a communication between Alice and Bob over an imperfect wire-tapping channel.

Let us assume that while Eve can observe all bits sent from Alice to Bob, she does so with a bit error probability, denoted by ε. To enhance security, Alice encodes each information bit (denoted as the N-th bit) by generating $N-1$ random bits and sending the bit as the XOR of these random bits and the information bit. Bob, who receives Alice's message error-free, can easily decode the information.

However, the probability of Eve correctly guessing the information bit becomes:

$$P_{\text{correct}} = \frac{1-(1-2\varepsilon)^N}{2},$$

which rapidly approaches $\frac{1}{2}$ as N increases.

Wyner's model was later extended by Csiszár and Körner [20], who examined a more general discrete memoryless broadcast channel. In this setup, Eve's received message is not necessarily a degraded version of the one Bob receives.

The input to both the main and wire-tapper channels is a random variable X, selected by Alice based on a probability distribution P_X. Bob and Eve receive the random variables Y and Z, respectively, which may take values from finite or countably infinite sets. The behavior of the channel is fully determined by the conditional probability distribution $P_{YZ|X}$.

In Wyner's original model, X, Y, and Z form a Markov chain, i.e.,

$$P_{Z|XY} = P_{Z|Y},$$

implying that the mutual information between X and Z, conditioned on Y, is zero:

$$I(X;Z|Y) = 0.$$

The secrecy capacity $C_S(P_{YZ|X})$ of the described broadcast channel, with transition probability distribution $P_{YZ|X}$, was introduced as the maximum rate at which Alice can reliably send information to Bob [20], while ensuring that the amount of information Eve gathers remains negligibly small.

In essence, secrecy capacity defines how many bits per use of the channel Alice can transmit in absolute confidentiality. Csiszár and Körner proved that:

$$C_S(P_{YZ|X}) \geq \max_{P(X)} [H(X|Z) - H(X|Y)]$$

Where the inequality holds, it is satisfied with equality, except in a few rare and exceptional cases that are not of significant interest. If equality is achieved, the secrecy capacity becomes zero unless $I(X;Y) > I(X;Z)$ for some probability distribution P_X.

To demonstrate that feedback from Bob to Alice over an insecure public channel can increase the secrecy capacity of a broadcast channel, we consider a scenario where both the main channel (Bob's) and Eve's channel are independent binary symmetric channels. These channels have respective bit error probabilities ϵ and δ, meaning that X, Y, and Z are binary random variables. The joint transition probability distribution is given by

$$P_{YZ|X} = P_{Y|X} \cdot P_{Z|X},$$

where

$$P_{Y|X}(y|x) = \begin{cases} 1-\epsilon & \text{if } x = y, \\ \epsilon & \text{if } x \neq y, \end{cases}$$

$$P_{Z|X}(z|x) = \begin{cases} 1-\delta & \text{if } x = z, \\ \delta & \text{if } x \neq z. \end{cases}$$

Without loss of generality, we may assume that $\epsilon \leq \frac{1}{2}$ and $\delta \leq \frac{1}{2}$. This setup models how feedback, even over an insecure public channel, can enhance the secrecy capacity by exploiting the error characteristics of both channels. To simplify notation, we will refer to the described probability distribution $P_{YZ|X}$ as $D(\epsilon, \delta)$. The secrecy capacity is given by:

$$C_S(D(\epsilon,\delta)) = \begin{cases} h(\delta) - h(\epsilon) & \text{if } \delta > \epsilon, \\ 0 & \text{otherwise,} \end{cases}$$

where $h(\cdot)$ denotes the binary entropy function.

It follows that secret messages can only be sent if $\delta > \epsilon$. However, if public feedback is allowed, secret communication can still occur even when $\delta < \epsilon$. To achieve this, Bob can conceptually reverse the channel scenario, making himself the sender while Alice and Eve become the receivers [21]. Alice first sends a random bit over the noisy broadcast channel to both Bob and Eve. Bob then XORs the bit he wants to send to Alice with the received random bit and transmits the result over the public channel. Alice can easily recover Bob's bit by XORing it with the random bit she initially sent, allowing her to receive the message with an error probability of ϵ. Meanwhile, Eve perceives Bob's bit as if it were transmitted through a combination of the two noisy channels, resulting in an error probability of $\epsilon + \delta - 2\epsilon\delta$ for her. Thus, the secrecy capacity with feedback is given by

$$h(\epsilon + \delta - 2\epsilon\delta) - h(\epsilon),$$

which is positive unless $\epsilon = \frac{1}{2}$ or $\delta = 0$.

V. Entropy

Entropy is a measure of the uncertainty or information content in a message or system [3], [2]. In the context of information theory, the entropy $H(X)$ of a discrete random variable X with possible outcomes $x_1, x_2, \ldots, x_n$, and corresponding probabilities $p(x_1), p(x_2), \ldots, p(x_n)$ is defined as:

$$H(X) = -\sum_{i=1}^{n} P_X(x_i) \log_2 P_X(x_i)$$

where:

- $P(x_i)$ is the probability of symbol x_i in the dataset.
- $\log_2$ is the logarithm base 2, since entropy is measured in bits.
- x_i are the individual possible values of X (e.g., unique byte values).
- X is the random variable representing the key's possible values (byte values in our case).

A. Calculation of entropy with a generated key

The process of calculating entropy for a generated key is crucial in assessing the randomness and unpredictability of encryption methods. For this analysis, an encrypted key generated using the AES algorithm on the input text "I love cryptography" is used. The resulting key is represented in hexadecimal format as follows:

$$\text{Encrypted Key: } 9a4c42f104b2bd82ef0a8fbce4440d$$

There are certain steps that need to be followed for calculating the entropy:

Step-1: Determine the Symbol Set: Identify the set of unique symbols (or bytes in the case of binary data) from the provided key.

$$\text{Symbol Set: } 9a, af, 4c, 42, f1, 04, b2, bd, 82,$$

$$ef, 0a, 8f, bc, 65, 44, 0d$$

Step-2: Count the frequency of each symbol. Each symbol has occurred once, so the frequency of each symbol is 1. Therefore, the frequency of each symbol is:

$$\text{Frequency of each symbol} = 1$$

Step-3: Calculate the Probability of Each Symbol. The probability $P_X(x_i)$ of each byte is the frequency of that byte divided by the total number of bytes (i.e., 16 for this case). Since each byte appears once, the probability for each byte is:

$$P_X(x_i) = \frac{1}{16} = 0.0625$$

Step-4: Apply the Entropy Formula. The entropy $H(X)$ is given by the formula:

$$H(X) = -\sum_{i=1}^{n} P_X(x_i) \log_2 P_X(x_i)$$

Since all probabilities are the same, we can simplify this formula:

$$H(X) = -16 \times \left(\frac{1}{16} \times (-4)\right) = -16 \times \frac{-4}{16} = 4$$

The entropy is 4 bits per symbol, as shown mathematically. This means there is a moderate level of randomness in the encrypted key.

Similarly, we can calculate the entropy of all the encrypted keys mathematically once we are well familiar with the hexadecimal symbols and bytes that are present in a key.

B. Key Findings

1) Symmetric Algorithm: The results presented showcase a comparison of key entropy values across four popular symmetric encryption algorithms: AES, 3DES, DES, and Blowfish. Here's a brief review of the findings:

Ref. text for encryption: "I love cryptography"

- **AES (Advanced Encryption Standard):** AES encryption uses a 128-bit key, and its key entropy was calculated to be 4.0000 bits. AES is widely known for its security and efficiency in various applications.
- **3DES (Triple DES):** 3DES uses a key length of 192 bits, resulting in the highest entropy among the compared algorithms, and its key entropy was calculated to be 4.3035 bits. Despite its higher security, 3DES is generally slower and less efficient than AES due to its triple encryption process.
- **DES (Data Encryption Standard):** DES, employing a 56-bit key, has the lowest key entropy, making it more vulnerable to brute-force attacks, and its key entropy was calculated to be 3.0000 bits. Its use has significantly declined due to these security limitations.
- **Blowfish:** Blowfish is a flexible algorithm that can use key lengths from 32 to 448 bits, and its key entropy was calculated to be 3.8750 bits. In this experiment, it shows a relatively high entropy compared to DES but falls slightly below AES and 3DES.

In this comparison, the key entropy was calculated for different symmetric algorithms to understand their randomness and security levels. 3DES exhibited the highest key entropy, followed closely by AES and Blowfish. DES, being an older encryption standard, demonstrated the lowest entropy, confirming its reduced security in modern contexts. These findings reinforce the use of AES and Blowfish in current encryption standards, whereas DES has been largely deprecated.

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

2) Asymmetric Algorithm Key Entropy: Asymmetric cryptographic algorithms play a crucial role in secure communications, especially in key exchange, digital signatures, and encryption. Here, we compare the entropy of private and public keys for three commonly used algorithms: RSA, DSA, and

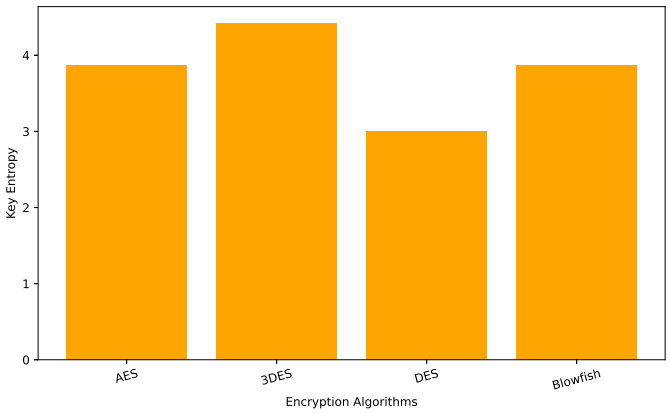

Fig. 2. Key Entropy for different encryption algorithm

ECC. Entropy is a measure of randomness and indicates the security strength of keys generated by these algorithms.

- **RSA (Rivest-Shamir-Adleman):** RSA, a widely used algorithm for encryption and digital signatures, relies on the difficulty of factoring large prime numbers. The entropy of the generated private key is 6.0, and the entropy of the generated public key is 5.8. RSA provides relatively high entropy for both private and public keys, with a slightly higher entropy for private keys due to their longer length and structure.
- **DSA (Digital Signature Algorithm):** DSA is primarily used for digital signatures, ensuring message integrity and authenticity. The entropy of the generated private key is 6.0, and the entropy of the generated public key is 5.9. Similar to RSA, DSA also shows high entropy for both key types. Its private key entropy is slightly higher due to the sensitive nature of key signing in the algorithm.
- **ECC (Elliptic Curve Cryptography):** ECC offers similar levels of security as RSA and DSA but with smaller key sizes, making it more efficient for mobile and constrained environments. The entropy of the generated private key is 5.9, and the entropy of the generated public key is 5.7. ECC shows slightly lower entropy than RSA and DSA due to its more compact key structure, but still offers high security with smaller keys.
  From above algorithms we can conclude:
  - RSA and DSA have nearly identical entropy values for private and public keys, which reflects their reliance on large key sizes and complex mathematical operations.
  - ECC, while having slightly lower entropy, provides a more efficient solution with smaller key sizes without compromising much on security.
  - In general, higher entropy correlates with more robust cryptographic strength, and all three algorithms perform well, with slight trade-offs between key size and efficiency.
  - This analysis highlights the balance between key strength (entropy) and algorithmic efficiency, making RSA preferable in resource-constrained environments, while ECC and DSA remain widely used in general cryptographic applications.

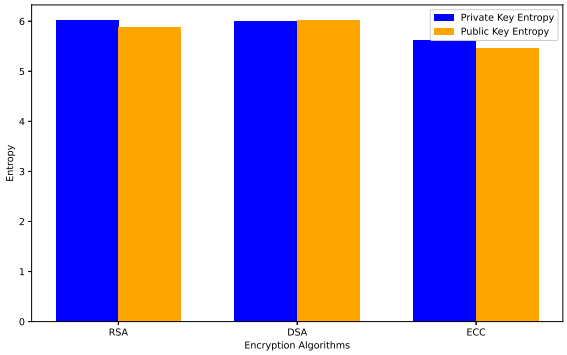

Fig. 3. Key entropy table for different asymmetric algorithms

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

REFERENCES

[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. 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 optimised 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.