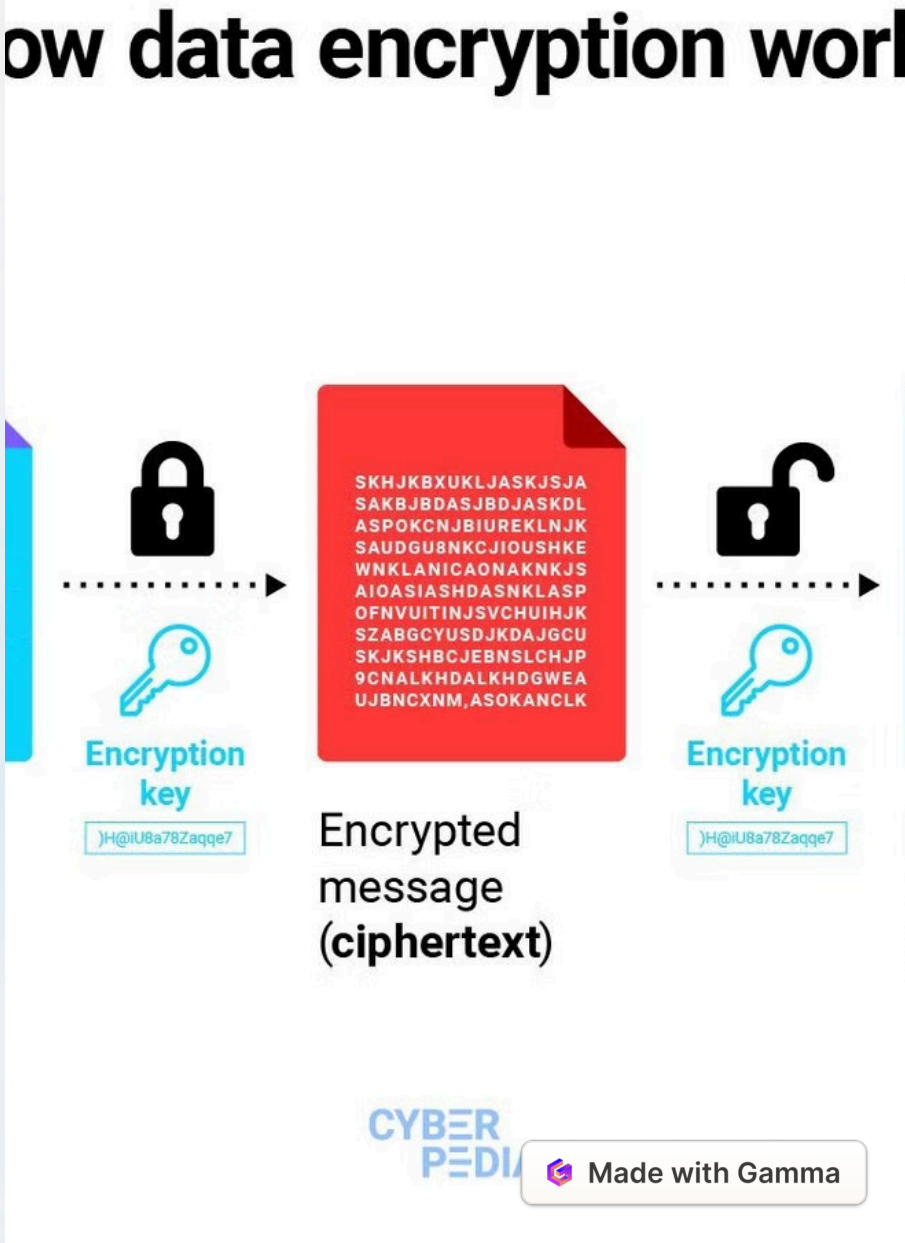
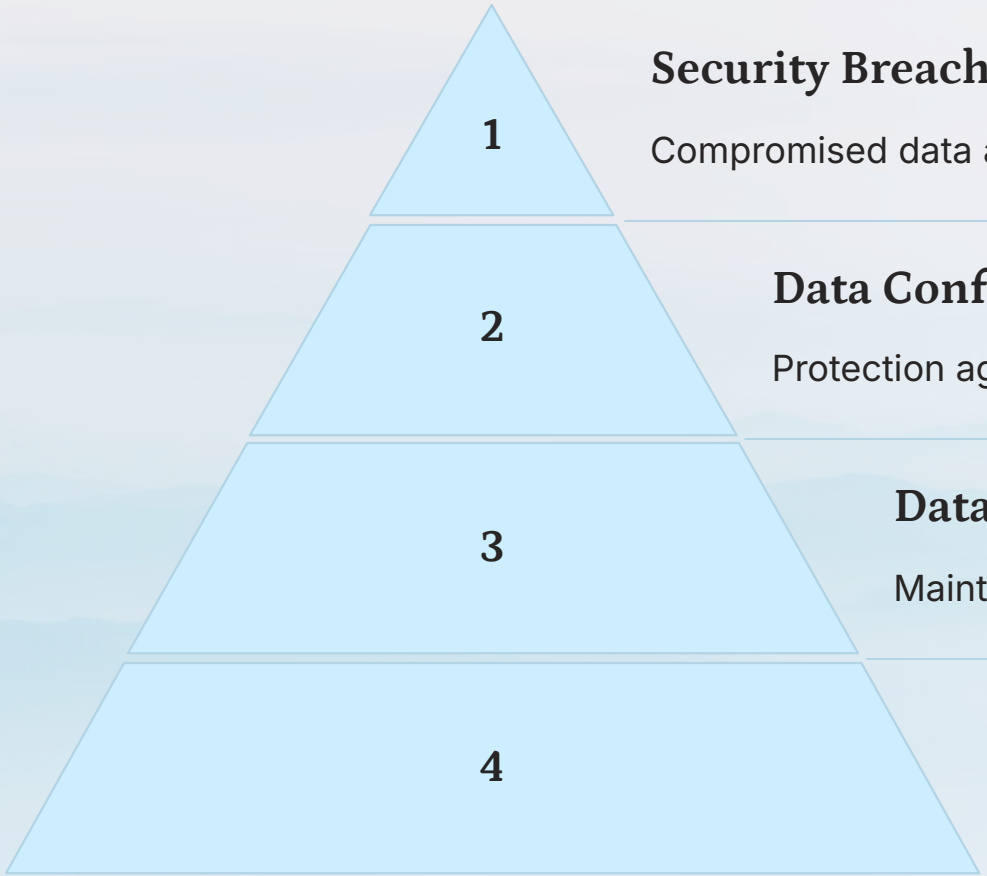
Encryption and Decryption Using Genetic Algorithms

This presentation will explore the application of genetic algorithms in cybersecurity for encrypting and decrypting sensitive data, including images and text.



Problem Definition and Specification



Security Breaches

Compromised data and systems

Data Confidentiality

Protection against unauthorized access

Data Integrity

Maintaining data accuracy and reliability

Data Availability

Ensuring data accessibility when needed

The problem involves safeguarding sensitive information from malicious actors. This requires addressing various security challenges, including data confidentiality, integrity, and availability. The goal is to develop robust encryption and decryption techniques for image and text data to ensure secure data transmission and storage.

Literature Survey

1 Genetic Algorithms in Cryptography

Numerous research papers have explored the application of genetic algorithms to cryptographic challenges, including key generation, cryptanalysis, and secure communication.

2 Image and Text Encryption

Studies have investigated the use of genetic algorithms for encrypting and decrypting images and text data, focusing on enhancing security and efficiency.

3 Hybrid Approaches

Researchers have combined genetic algorithms with traditional cryptographic techniques to create robust encryption methods that leverage the strengths of both.

4 Performance and Security Analysis

Extensive research has been conducted to assess the performance and security of genetic algorithm-based encryption schemes, comparing them to existing methods.

Broad Knowledge of Available Techniques to Solve a Particular Problem

Encryption Techniques

Several encryption techniques are available to secure data. Symmetric encryption, like AES, uses the same key for encryption and decryption. Asymmetric encryption, like RSA, uses separate keys for each process.

Genetic Algorithms

Genetic algorithms are a type of evolutionary computing technique inspired by biological evolution. They use a population of candidate solutions to optimize a problem by mimicking natural selection.

Image and Text Encoding

Image and text encoding methods, such as Huffman coding or Run-Length Encoding, can be used to reduce the size of data before encryption.

Security Considerations

Security considerations are essential, including key management, vulnerability analysis, and potential attacks like brute-force attacks or side-channel attacks.

Planning of the Work

1

Project Initiation

Define project scope, goals, and resources. Establish a clear project plan and timeline.

2

Requirements Analysis

Identify specific requirements for encryption, decryption, and genetic algorithm implementation.

3

System Design

Choose appropriate encryption algorithms, genetic algorithm parameters, and data encoding techniques.

4

Implementation

Develop code modules for encryption, decryption, genetic algorithm optimization, and data encoding.

5

Testing

Thoroughly test the system's security, efficiency, and accuracy against various data sets and attack scenarios.

6

Deployment and Maintenance

Deploy the system into a production environment and ensure ongoing security updates and performance monitoring.

CyberArk Cybersecurity Predictions

2024

40%
of all cyberattacks will be linked to session hijacking

30%
of organizations will experience increased data breaches due to credential theft

55%
of enterprises will expedite tech consolidation to simplify security

2025

80%
of organizations will fail to protect AI-driven security mechanisms, fueling a vicious cyber risk cycle

60%
of Fortune 2000 company CISOs will champion transparent, rapid disclosure practices

2026

45%
of Fortune 500 company boards will seek out a chief AI security officer

60%
of all regulated global entities will struggle to comply with ever-increasing data protection and breach disclosure requirements

Major global powers
will call for a Cybersecurity Geneva Convention

Genetic Algorithm Fundamentals

Darwinian Inspiration

Genetic algorithms are inspired by natural selection and evolution. They use principles of survival of the fittest and genetic recombination to solve complex problems.

Population of Solutions

A genetic algorithm starts with a population of potential solutions, represented as chromosomes or individuals. Each individual is a possible solution to the problem.

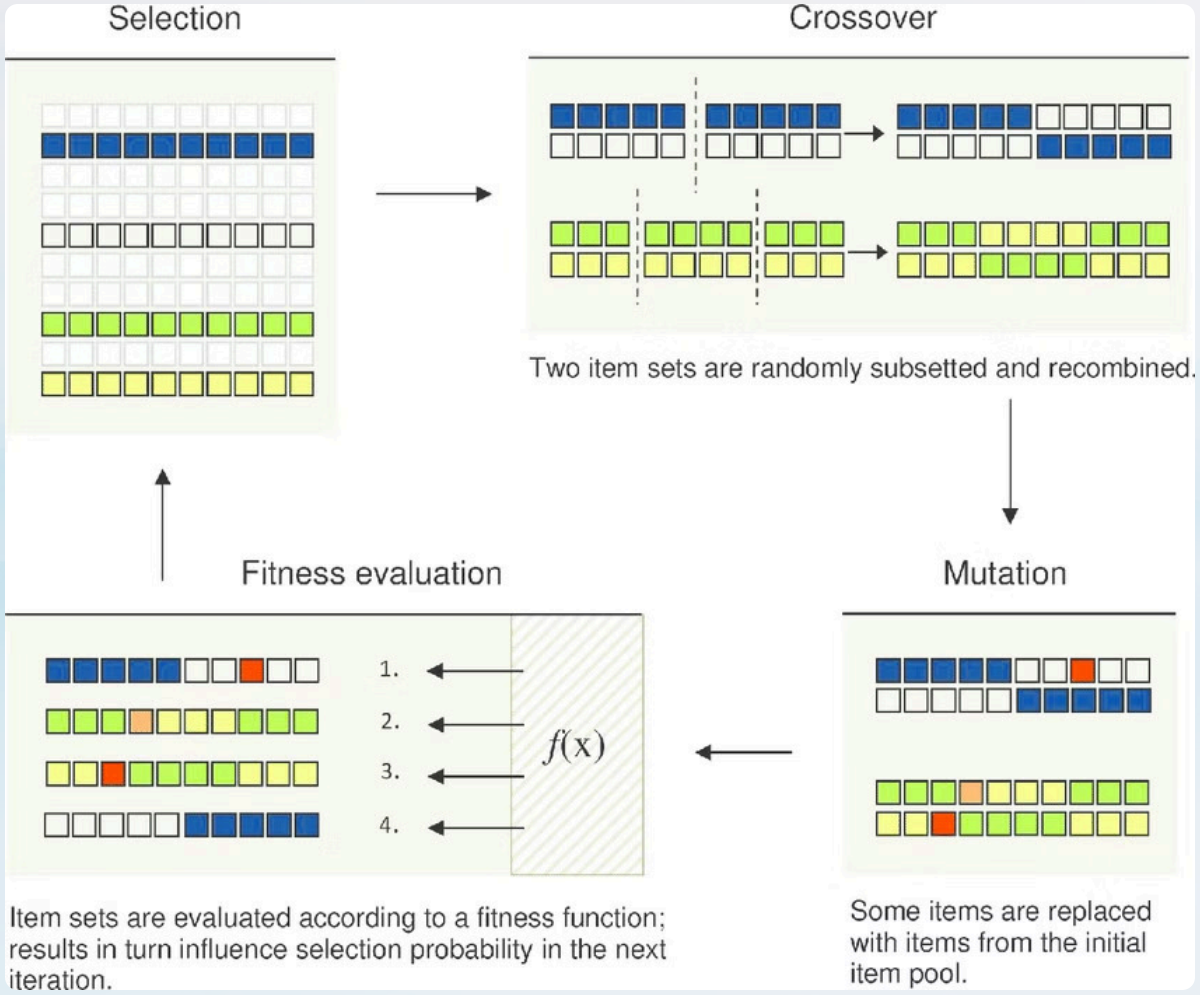
Fitness Evaluation

The fitness of each individual is evaluated based on how well it solves the problem. Higher fitness indicates a better solution. The algorithm then selects individuals based on their fitness scores for reproduction.

Reproduction and Mutation

Selected individuals reproduce to create offspring, inheriting traits from their parents. Mutations are introduced to introduce diversity and explore new solution spaces. This process continues over generations, with the population evolving towards better solutions.

Illustration of GA



How does Genetic Algorithim work?

1 Initialization

Start with a population of candidate solutions, represented as chromosomes or individuals.

2 Fitness Evaluation

Evaluate the fitness of each individual based on how well it solves the problem.

3 Selection

Select individuals with higher fitness scores to participate in reproduction.

4 Reproduction

Reproduce selected individuals to create new offspring, inheriting traits from parents.

5 Mutation

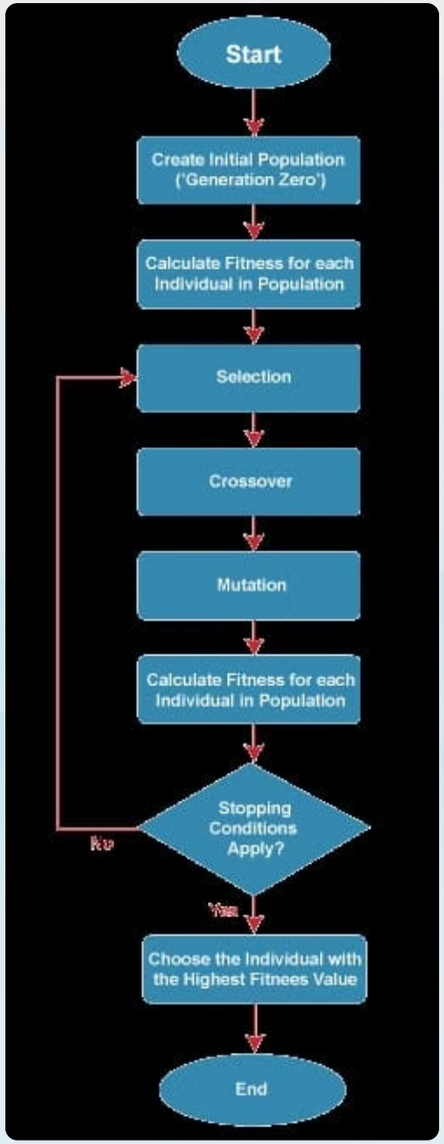
Introduce random mutations to the offspring to explore new solution spaces.

6 Iteration

Repeat the process over multiple generations until the optimal solution is found.

This cycle of fitness evaluation, selection, reproduction, and mutation continues over multiple generations, allowing the population to evolve towards better solutions.

Flow chart of GA



1. Initialization

Population is a subset of solutions in the current generation. It can also be defined as a set of chromosomes.The population is usually defined as a two dimensional array of – **size population, size x, chromosome size**.

Population Initialization

There are two primary methods to initialize a population in a GA. They are –

- **Random Initialization**
- **Heuristic Initialization**

Random initialization is a common method used to create the initial population in a genetic algorithm. This process involves generating random values for each gene in each individual in the population. The goal is to ensure that the initial population is diverse, covering a wide range of possible solutions. This diversity helps the genetic algorithm to effectively explore the solution space and avoid premature convergence to suboptimal solutions.

Linear Equality Problem

So here is the example of applications of genetic algorithm to solve the simple mathematical linear equality problem. Suppose there is equality a + 2b + 3c + 4d+5e = 20, genetic algorithm will be used to find the value of a, b, c, d and e that satisfy the above equation for this problem the objective is minimizing the value of function f(x) where f(x) = ((a + 2b + 3c + 4d+5e) - 20). Since there are five variables in the equation, namely a, b, c, d and e we can compose the chromosome as follow: To speed up the computation, we can restrict that the values of variables a, b, c, and d are integers between 0 and 20.

Step 1 : Initialisation

For example we define the number of chromosomes in population are 6, then we generate random value of gene a, b, c ,d and e for 6 chromosomes

Chromosome[1] = [a;b;c;d;e] = [11;05;04;00;08]

Chromosome[2] = [a;b;c;d;e] = [09;15;03;01;03]

Chromosome[3] = [a;b;c;d;e] = [04;00;01;08;02]

Chromosome[4] = [a;b;c;d;e] = [01;05;13;04;06]

Chromosome[5] = [a;b;c;d;e] = [02;09;12;09;02]

Chromosome[6] = [a;b;c;d;e] = [08;14;05;11;00]

2. Fitness evaluation

The fitness function simply defined is a function which takes a **candidate solution to the problem as input and produces as output** how “fit” our how “good” the solution is with respect to the problem in consideration.

A fitness function should possess the following characteristics –

- The fitness function should be sufficiently fast to compute.
- It must quantitatively measure how fit a given solution is or how fit individuals can be produced from the given solution.

The following image shows the fitness calculation for a solution of the 0/1 Knapsack. It is a simple fitness function which just sums the profit values of the items being picked (which have a 1), scanning the elements from left to right till the knapsack is full.

Step 2:Evaluation

We calculate the objective function value for each chromosome produced in the initialization step:

$$\mathbf{F\_obj[1]} = \text{Abs}((11 + 2*5 + 3*4 + 4*0 + 5*8) - 20) = 53$$

$$\mathbf{F\_obj[2]} = \text{Abs}((9 + 2*15 + 3*3 + 4*1 + 5*3) - 20) = 47$$

$$\mathbf{F\_obj[3]} = \text{Abs}((4 + 2*0 + 3*1 + 4*8 + 5*2) - 20) = 29$$

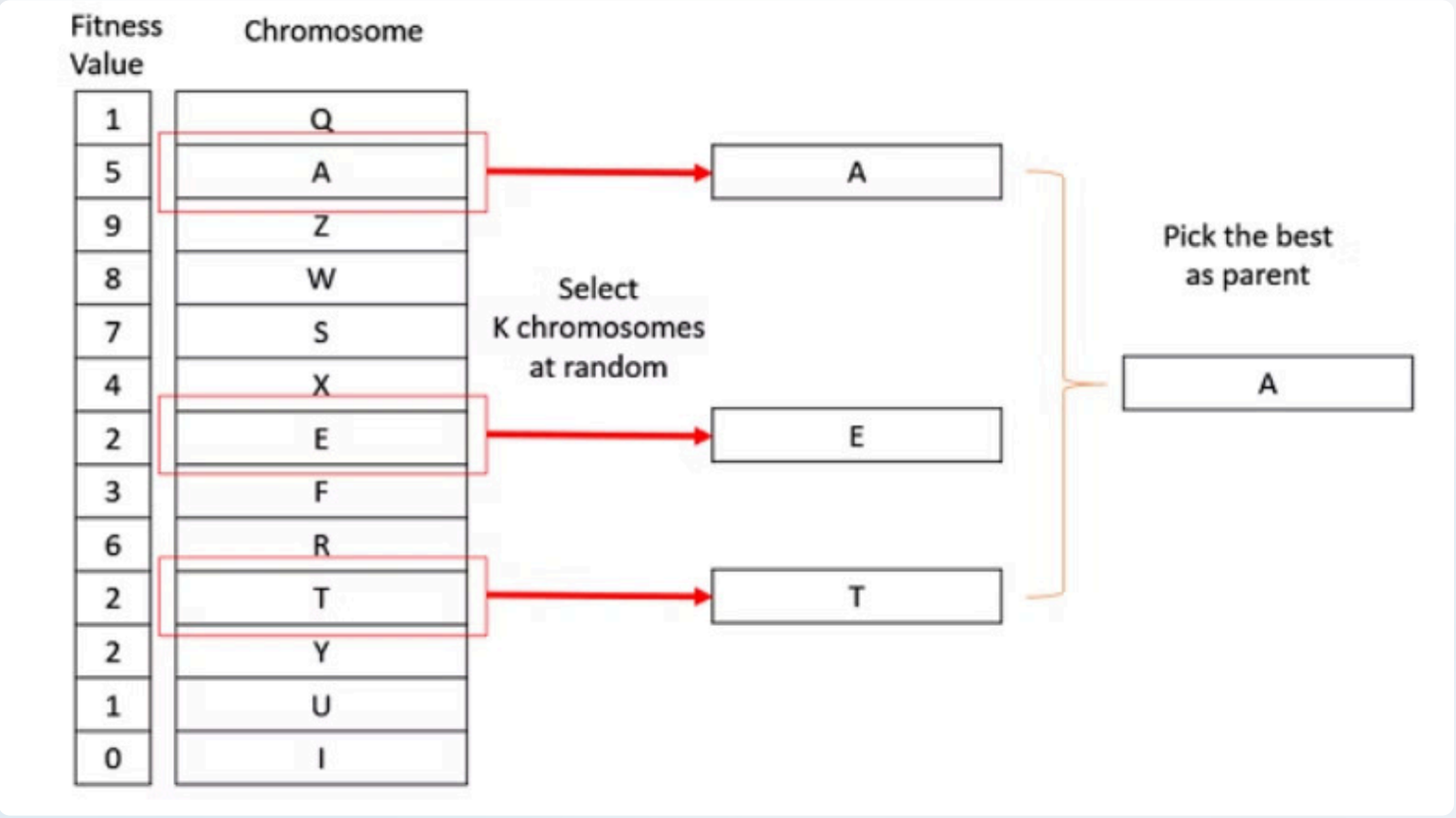
$$\mathbf{F\_obj[4]} = \text{Abs}((1 + 2*5 + 3*13 + 4*4 + 5*6) - 20) = 76$$

$$\mathbf{F\_obj[5]} = \text{Abs}((2 + 2*9 + 3*12 + 4*9 + 5*2) - 20) = 82$$

$$\mathbf{F\_obj[6]} = \text{Abs}((8 + 2*14 + 3*5 + 4*11 + 5*0) - 20) = 75$$

3. Tournament Selection

In K-Way tournament selection, we select K individuals from the population at random and select the best out of these to become a parent. The same process is repeated for selecting the next parent. Tournament Selection is also extremely popular in literature as it can even work with negative fitness values.



Step 3:Selection

The chromosomes which higher fitness will have more probability to be selected for the next generation. To calculate fitness probability we have to calculate the fitness of each chromosome. To avoid the divide by zero problem, the value of F_obj is added by 1 so that fitness will be calculated.

Fitness[1] = 1 / (1+**F_obj[1]**) = 1/(1+53) = 0.0185

Fitness[2] = 1 / (1+**F_obj[2]**) = 1/(1+47) = 0.0208

Fitness[3] = 1 / (1+**F_obj[3]**) = 1/(1+29)= 0.0333

Fitness[4] = 1 / (1+**F_obj[4]**) = 1/(1+76) = 0.013

Fitness[5] = 1 / (1+**F_obj[5]**) = 1/(1+82) = 0.0120

Fitness[6] = 1 / (1+**F_obj[6]**) = 1/(1+75) = 0.0132

Total=0.0185 + 0.0208 + 0.0333 + 0.013 + 0.0120+ 0.0132

=0.1108

(i)

P[i] = Fitness[i] / Total

P[1] = 0.0185 / 0.1108= 0.167

P[2] = 0.0208 / 0.1108 = 0.188

P[3] = 0.0333 / 0.1108 = 0.301

P[4] = 0.013 / 0.1108= 0.1173

P[5] = 0.0120 / 0.1108 = 0.1083

P[6] = 0.0132 / 0.1108 =0.1191

(ii)

Based on the probabilities, Chromosome 3 has the highest fitness and therefore the highest chance of selection for the next generation using the roulette wheel method. The cumulative probabilities should sum to 1; any deviation indicates computation errors.

C[1]=0.0167

C[2]= 0.0167 + 0.188=0.2606

C[3]= 0.0167 + 0.188 + 0.301=0.3518

C[4]= 0.0167 + 0.188 + 0.301+0.1173 =0.7167

C[5]= 0.0167 + 0.188 + 0.301+0.1173 + 0.1083 =0.7811

C[6]= 00.0167 + 0.188 + 0.301+0.1173 + 0.1083 + 0.1191=1.00

This process is used to generate random number R in the range 0-1 as follows.

R[1] = 0.209

R[2] = 0.482

R[3] = 0.111

R[4] = 0.842

R[5] = 0.589

R[6] = 0.801

(iii)

So here we do the comparison and on the basis of that new chromosomes will be formed. If random number R[1] is greater than C[1] and smaller than C[2] then select Chromosome[2] as a chromosome in the new population for next generation[1][2]:

NewChromosome[1] = Chromosome[2]

NewChromosome[2] = Chromosome[3]

NewChromosome[3] = Chromosome[1]

NewChromosome[4] = Chromosome[6]

NewChromosome[5] = Chromosome[3]

NewChromosome[6] = Chromosome[5]

Chromosomes in the population thus became:

Chromosome[1] = [09;15;03;01;03]

Chromosome[2] = [04;00;01;08;02]

Chromosome[3] = [11;05;04;00;08]

Chromosome[4] = [08;14;05;11;00]

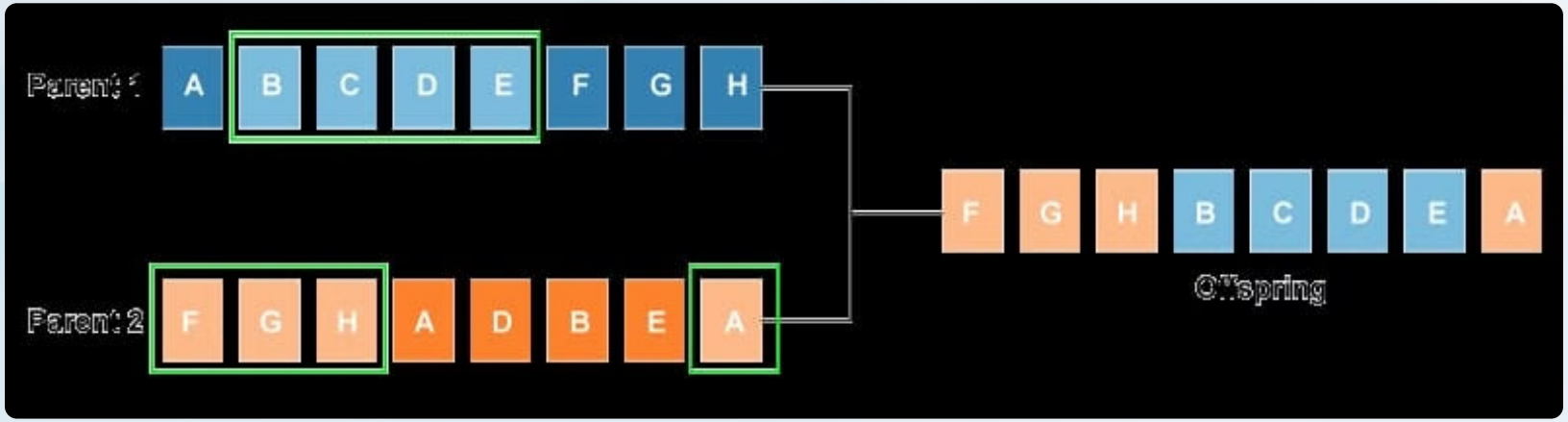
Chromosome[5] = [04;00;01;08;02]

Chromosome[6] = [02;09;12;09;01]

(iv)

4. Reproduction

Reproduction in genetic algorithms involves selecting individuals with higher fitness, combining their genetic material through crossover, introducing random changes via mutation, and generating offspring for the next generation. This iterative process continues until a predetermined number of generations is reached.



Step 4:Reproduction

In this example, we use a one-cut point crossover where a position in the parent chromosome is randomly selected. Sub-chromosomes are exchanged up to this point. The parent chromosome for mating is chosen randomly, and the number of mated chromosomes is controlled by the crossover rate (ρc) parameter. Here is the pseudo-code for the crossover process:

```
Begin
    k← 0;
    while(k<population)do
    R[k] = random(0-1);
    if(R[k]< ρc) then
        select Chromosome[k] as parent;
end;
k = k + 1;
```

(i)

```
end;
end;s
```

(ii)

Chromosome k will be selected as a parent if R[k]. Suppose we set that the crossover rate is 25%, then Chromosome number k will be selected for crossover if random generated value for Chromosome k below 0.25. The process is as follows: First we generate a random number R as the number of population

R[1] = 0.117

R[2] = 0.482

R[3] = 0.185

R[4] = 0.199

R[5] = 0.356

R[6] = 0.892

For random number **R** above, parents are **Chromosome[1], Chromosome[3]** and **Chromosome[4]** will be selected for crossover.

Chromosome[1] >< Chromosome[3]

Chromosome[3] >< Chromosome[4]

Chromosome[4] >< Chromosome[1]

(iii)

Step 4:Reproduction

After selecting chromosomes, the next step is determining the crossover point. Random numbers between 1 and the chromosome length minus one are generated. For example, with a chromosome length of 5, the crossover point is 4. Genes are then exchanged at this point between parent chromosomes. For instance, if we generate 3 random numbers, we get:

C[1] = 2

C[2] = 3

C[3] = 1

(iv)

Then for first crossover, second crossover and third crossover, parent's gens will be cut at gen number 2, gen number 3 and gen number 1 respectively, e.g

Chromosome[1] = Chromosome[1] >< Chromosome[3]

= [09;15;03;01;03] >< [11;05;04;00;08]

= [09;15;04;00;08]

Chromosome[3] = Chromosome[3] >< Chromosome[4]

= [11;05;04;00;08] >< [08;14;05;11;00]

= [11;05;04;11;00]

Chromosome[4] = Chromosome[4] >< Chromosome[5]

= [08;14;05;11;00] >< [04;00;01;08;02]

= [08;00;01;08;02]

Thus Chromosome population after experiencing a crossover process:

Chromosome[1] = [09;15;04;00;08]

Chromosome[2] = [04;00;01;08;02]

Chromosome[3] = [11;05;04;11;00]

Chromosome[4] = [08;00;01;08;02]

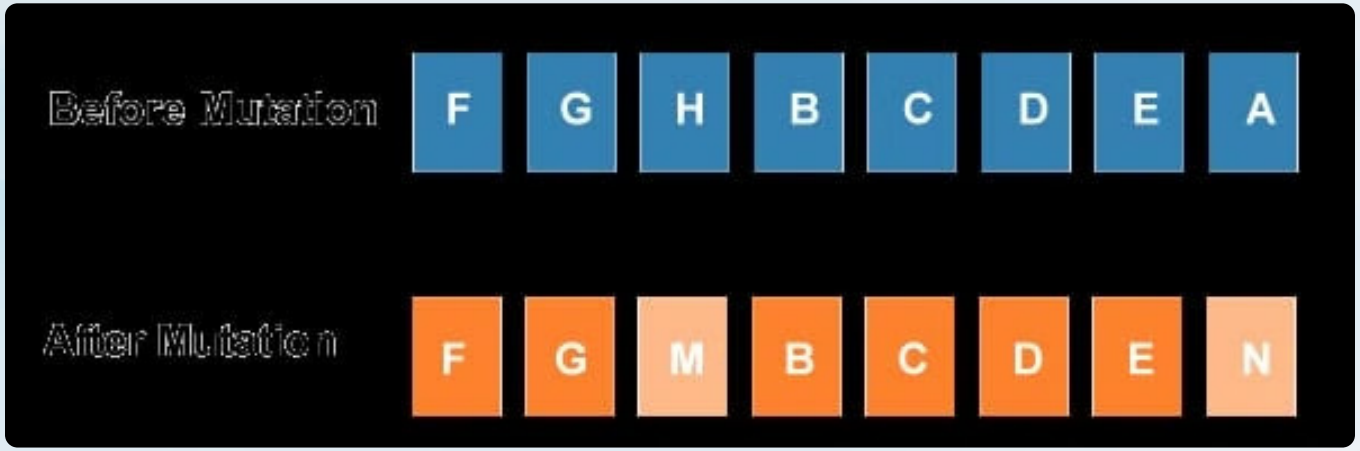
Chromosome[5] = [04;00;01;08;02]

Chromosome[6] = [02;09;12;09;01]

(v)

5. Mutation

Mutation in genetic algorithms involves making small, random changes to individual chromosomes within a population. It helps introduce genetic diversity, preventing the algorithm from getting stuck in local optima and allowing exploration of new solutions in the search space.



Step 5:Mutation

So, first we have to calculate the total length of generation in the population. The total length of the generation is

total_generation = number_of_generation_in_Chromosome * number of population

= 5 * 6 = 30

(i)

If the random numbers generated are 2, 14, and 28, the mutations occur at Chromosome 1, gene 2; Chromosome 3, gene 4; and Chromosome 6, gene 3. The values at these points are replaced by random numbers between 0 and 20. Suppose the generated random numbers are 2, 0, and 5. The chromosome composition after mutation is:

Chromosome[1] = [09;02;04;00;08]

Chromosome[2] = [04;00;01;08;02]

Chromosome[3] = [11;05;04;00;00]

Chromosome[4] = [08;00;01;08;02]

Chromosome[5] = [04;00;01;08;02]

Chromosome[6] = [02;09;05;09;01]

(ii)

After finishing the mutation process then we have one iteration or one generation of the genetic algorithm. Now, we can evaluate the objective function after one generation:

Chromosome[1] = [09;02;04;00;08]

F_obj[1] = Abs((9+ 2*2 + 3*4 + 4*0 + 5*8)-20)

= 45

F_obj[2] = Abs((4+ 2*0 + 3*1 + 4*8 +5*2)-20)

= 29

F_obj[3] = Abs((11+ 2*5 + 3*4 + 4*0 +5*0)-20)

= 13

F_obj[4] = Abs((8+ 2*0 + 3*1 + 4*8 +5*2)-20)

= 33

F_obj[5] = Abs((4+ 2*0 + 3*1 + 4*8 +5*2)-20)

= 29

F_obj[6] = Abs((2+ 2*9 + 3*5 + 4*9 +5*1)-20)

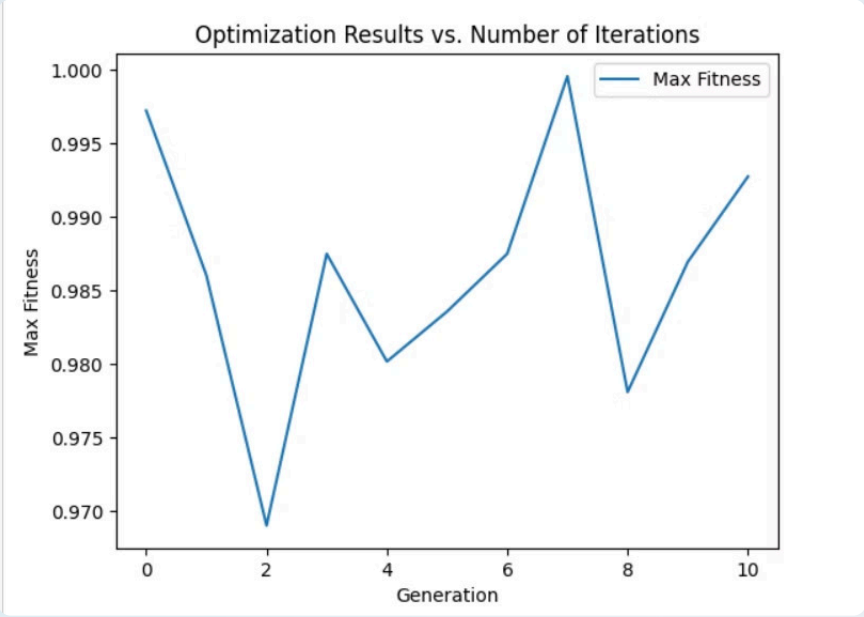
= 56

(iii)

These new chromosomes undergo the same genetic algorithm processes—evaluation, selection, crossover, and mutation—as the previous generation. This cycle repeats until a predetermined number of generations is reached.

6. Iteration

The genetic algorithm starts with a randomly generated initial population. It iterates through generations where individuals undergo selection, crossover, and mutation, mimicking natural evolution. Over multiple generations, the algorithm explores the solution space, favoring higher fitness values. This process helps converge towards an optimal or near-optimal solution. The cycle repeats until a stopping criterion, such as a maximum number of generations or satisfactory fitness, is met.



Step 6:Results

By executing the simple genetic algorithm code. The primary work of the GA is performed in three routines selection,crossover, and mutation we get the best values of the variables a, b, c, d and e.

SGA Parameters

Population size = 10

Chromosome length = 5

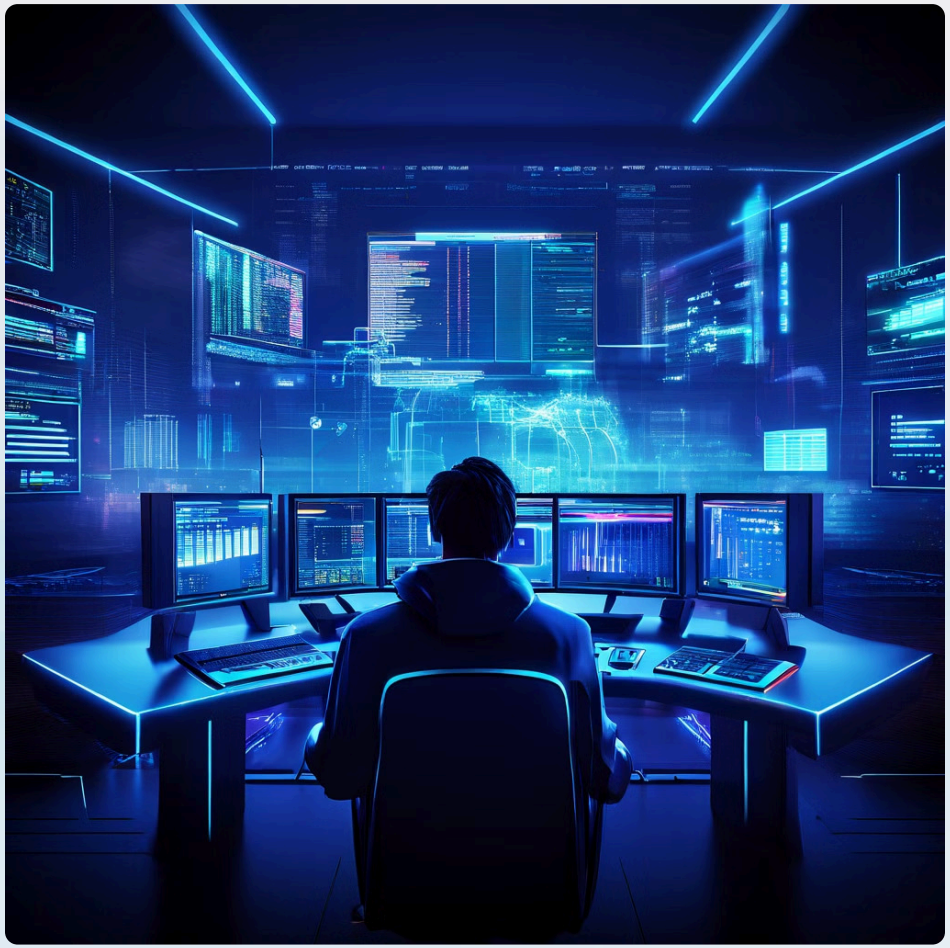
Maximum of generation = 20

Crossover probability = 0.25

Mutation probability = 0.01

Linear equation problem

A value	*B value*	*C value*	*D value*	*E value*
1	*1*	*3*	*2*	*0*
2	*6*	*4*	*0*	*0*
1	*0*	*2*	*2*	*1*
1	*1*	*0*	*3*	*1*
2	*0*	*1*	*0*	*3*
6	*0*	*3*	*0*	*1*
4	*4*	*0*	*2*	*0*
0	*3*	*2*	*2*	*0*
3	*2*	*2*	*0*	*1*
4	*4*	*0*	*2*	*0*



Cryptography: Securing Data

Cryptography is the practice of securing information through the use of codes, ciphers, and algorithms. It is a fundamental component of data security, ensuring the confidentiality, integrity, and authenticity of digital communications and information.

Cryptographic techniques are used to protect sensitive data from unauthorized access, manipulation, or disclosure. They play a crucial role in securing online transactions, protecting personal information, and safeguarding critical infrastructure.

Encryption

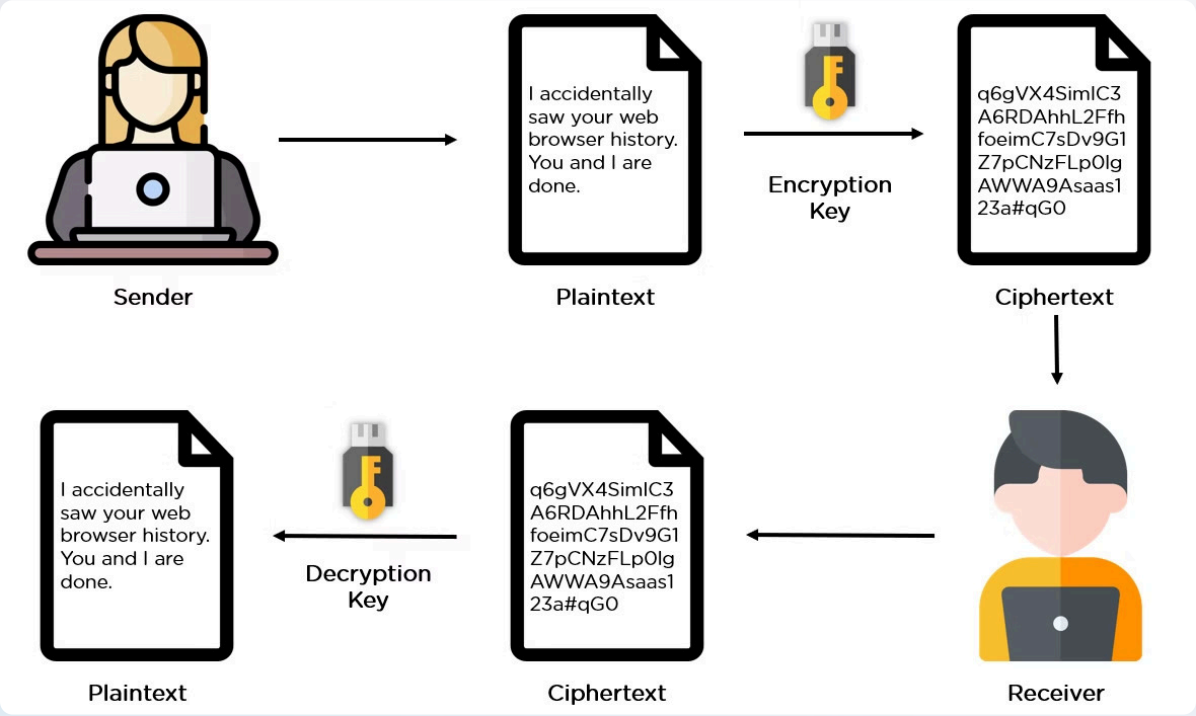
Encryption helps us to secure data that we send, receive, and store. It can consist text messages saved on our cell-phone, logs stored on our fitness watch, and details of banking sent by your online account.

It is the way that can climb readable words so that the individual who has the secret access code, or decryption key can easily read it. For diplomatic information to help in providing data security.

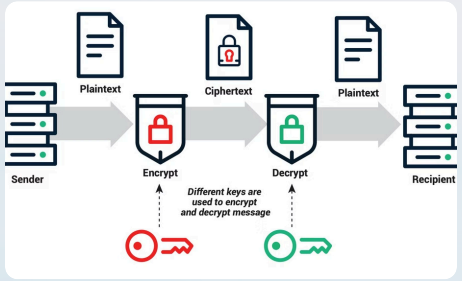
Decryption: Unlocking the Encrypted Data

Decryption is the process of converting encrypted data back into its original, readable form. This is achieved through the use of a decryption key, which is typically a password or a secret code that reverses the encryption process. By applying the correct decryption key, the encrypted data can be accessed and understood by the intended recipient. Decryption is a crucial step in ensuring the security and privacy of sensitive information, enabling authorized parties to securely access and utilize the data as needed.

Iilustrating Cryptography

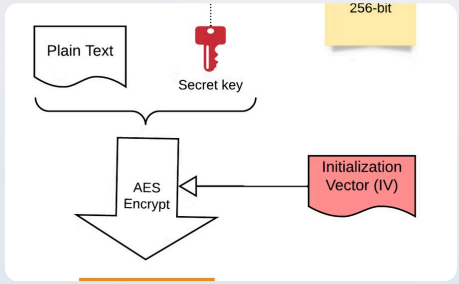


Encryption and Decryption Algorithms



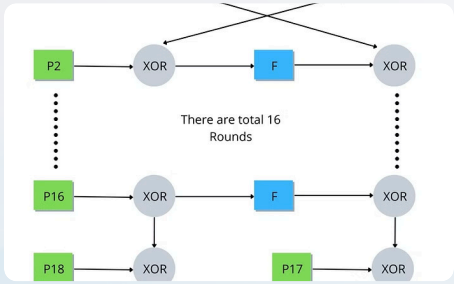
RSA Algorithm

RSA is a popular asymmetric encryption algorithm. It uses two keys, a public key and a private key.



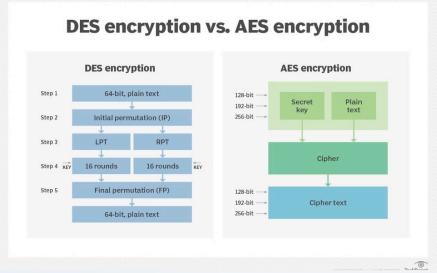
AES Algorithm

AES is a symmetric encryption algorithm used for data encryption. It uses a single key for both encryption and decryption.



Blowfish Algorithm

Blowfish is a symmetric block cipher. It operates on 64-bit blocks and uses a variable length key up to 448 bits.



Twofish Algorithm

Twofish is a symmetric block cipher algorithm. It is a Feistel cipher that operates on 128-bit blocks

XOR Cipher

The XOR (Exclusive OR) operation is a fundamental cryptographic technique used for both encryption and decryption. It works by applying a bitwise XOR operation between the plaintext and a secret key to produce the ciphertext, and then repeating the same operation to decrypt the message.To encrypt a message using the XOR cipher, the sender first converts the plaintext into a binary representation. They then generate a random key of the same length as the plaintext, also in binary form.
For example, if the plaintext is "1010" and the key is "1100", the encryption process would be:

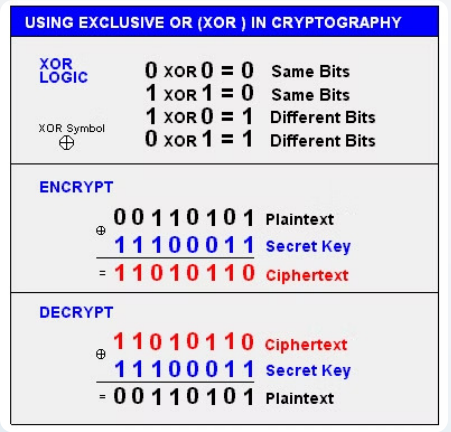
- Plaintext: 1010
- Key: 1100
- Ciphertext: 0110

To decrypt the message, the receiver simply repeats the XOR operation using the same key, reversing the encryption process and recovering the original plaintext.The XOR cipher is a simple but effective encryption method, as the ciphertext provides no information about the original plaintext without the secret key. It is commonly used in various applications, such as data protection, communication security, and computer networking.

The Working Principles

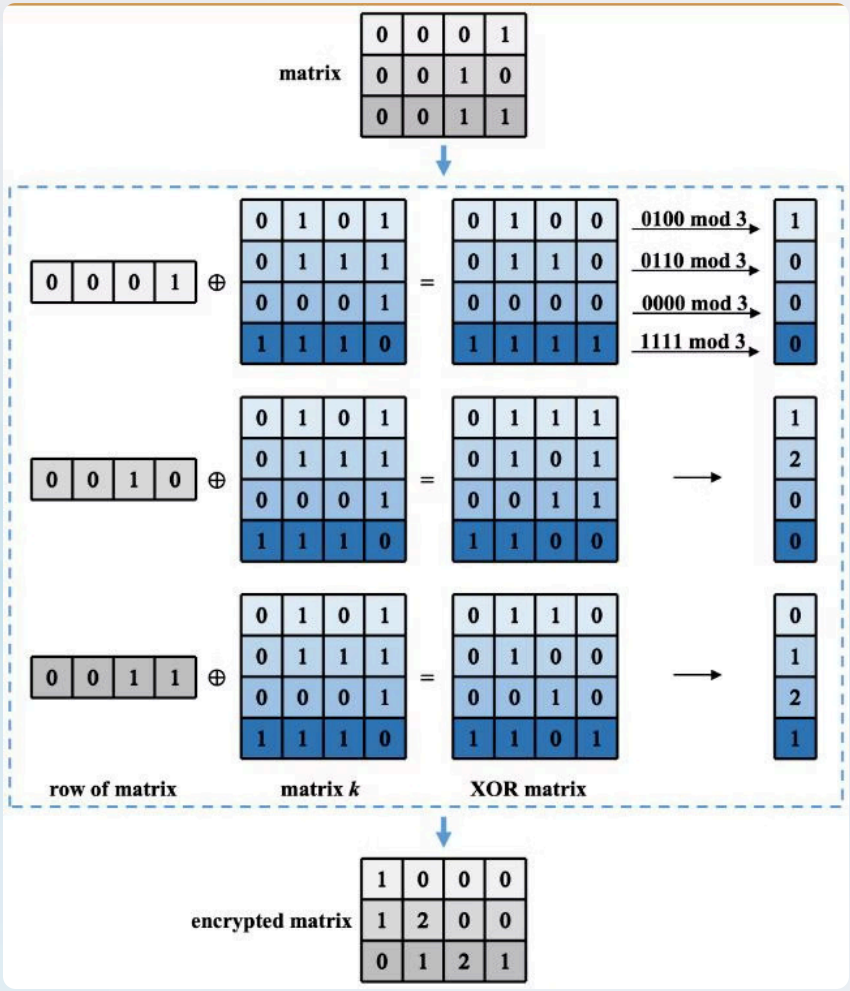
A	B	Q
0	0	0
1	0	1
0	1	1
1	1	0

XOR logic gate and truth table



- The decryption process is identical to the encryption process. By XORing the encrypted message with the same key, the original plaintext is recovered.
- This is because $(A \oplus B) \oplus B = A$

Example



Implementation and Experimental Evaluation

The proposed genetic algorithm for encryption and decryption will be implemented using a suitable programming language such as Python or Java. This implementation will involve the development of modules for encoding and decoding text and images using the selected encryption algorithm.

The performance of the genetic algorithm will be evaluated through extensive experiments. This will involve encrypting and decrypting various data sets of text and images with different sizes and complexity levels. The experimental evaluation will assess the algorithm's accuracy, speed, and security.

1

Accuracy

Percentage of correctly decrypted data.

2

Speed

Time required for encryption and decryption.

3

Security

Resistance to attacks and unauthorized access.

The experimental results will be analyzed to determine the effectiveness of the genetic algorithm and its suitability for real-world cybersecurity applications.

Image Evaluation

In the field of machine learning, two commonly used metrics for evaluating the performance of image processing and computer vision models are Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index (SSIM).

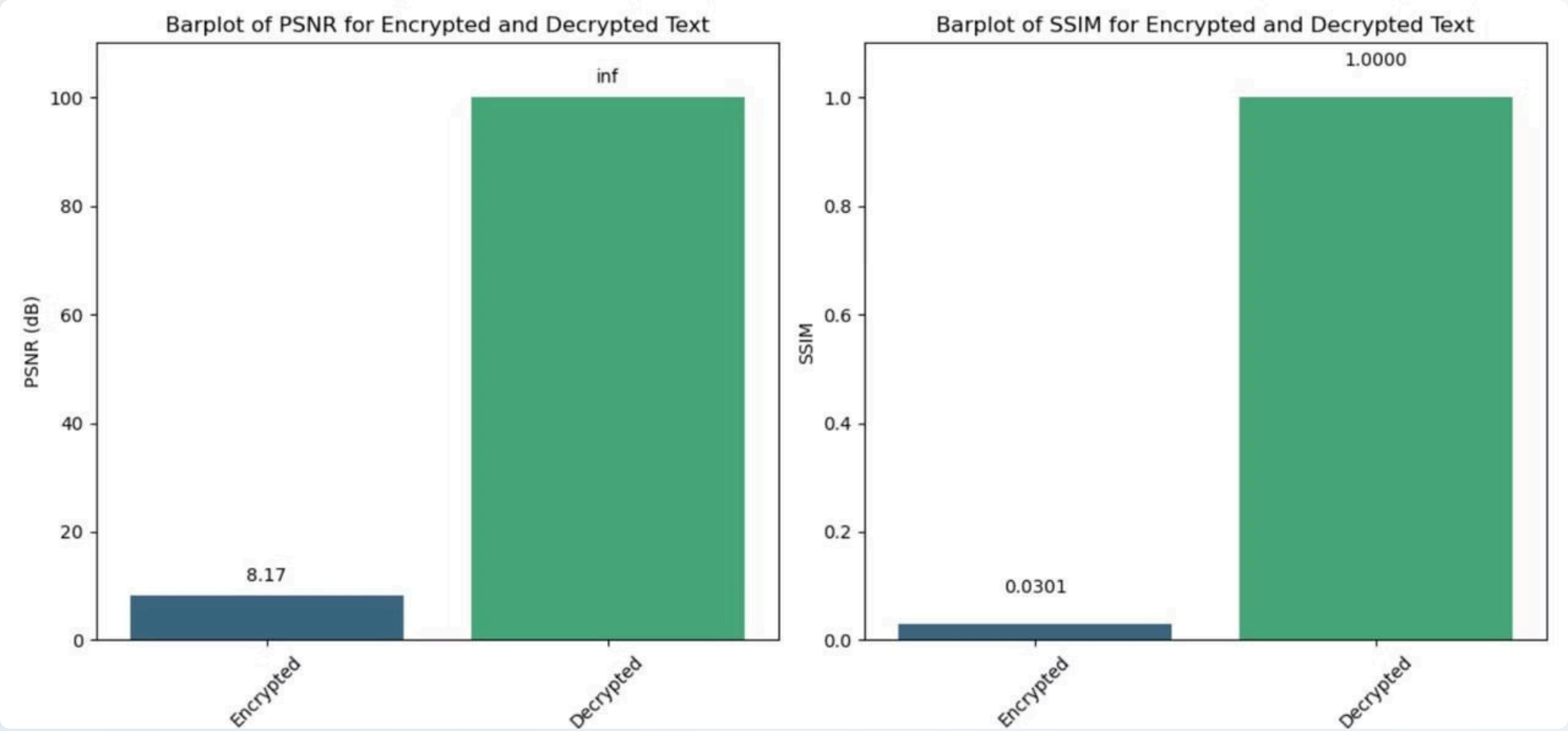
PSNR is a measure of the ratio between the maximum possible signal power and the noise power. It is often used to quantify the quality of reconstructed or processed images compared to the original. A higher PSNR value indicates better image quality.

Psnr = 20*log10(max_i) - 10* log10(mse) Max_i is max possible pixel value

SSIM, on the other hand, is a more sophisticated metric that takes into account the structural similarities between the original and processed images. SSIM can better capture perceptual differences that are relevant to the human visual system.

Both PSNR and SSIM play important roles in the evaluation and optimization of machine learning models that involve image-based tasks, such as image denoising, super-resolution, and segmentation.

Preparation of bar charts



Conclusion

This project showcases the successful use of a Genetic Algorithm (GA) to optimize encryption keys for text and image data, ensuring secure data transmission and storage. Preprocessing steps, such as converting images to grayscale and flattening them, streamline the data for GA processing. The model architecture covers input preprocessing, GA setup, execution, and postprocessing, resulting in effective encryption and decryption.

Key highlights include the GA's efficiency in handling large datasets, ensuring scalability, robustness in providing consistent and reliable results, and flexibility through customizable GA parameters. Applications span data security, digital watermarking, cryptographic research, and secure image transmission, demonstrating the practical significance of GAs in encryption tasks.

In conclusion, the project underscores the potential of genetic algorithms in generating robust encryption keys, providing a foundation for further research and development in secure data encryption techniques.