Messaging Web Application Using High-Speed Encryption Algorithm with Polynomial Roots

Team Members:

- Mohamed Ahmed Elsayed
- Mohamed Abdelfattah Ahmed
- Abdelfattah Zakaria Abdelfattah
- Shrouk Elsayed Mohamed
- Dalia Abdallah Mohamed
- Sara Reda Abdallah

Under supervision of
Dr. Sayed Badr

Agenda

- Problem Definition
- Project Idea
- Root Finding Algorithms
- How Does Our Encryption Algorithm Works
- Results
- Current Solutions in the Market
- Methodology
- Current Progress
- Future Work

Problem Definition

The current encryption algorithms are slow and consume a lot of energy, which is not suitable for real-time applications.
They can cause multiple problems like:

- Increased Processing Overhead.
- Impact on Real-Time Systems.
- Increased energy consumption and carbon footprint.

1- Increased Processing Overhead

- Slower encryption algorithms require more computational resources (such as CPU cycles and memory) to perform encryption and decryption operations. This increased processing overhead can strain system resources, leading to higher costs, reduced scalability, and potentially degraded user experience.

2-Impact on Real-Time Systems

- In real-time systems where timely processing is critical (e.g., communication systems, financial transactions), slow encryption algorithms may introduce unacceptable delays, affecting the responsiveness and reliability of the system.

3-Increased energy consumption and carbon footprint

- In large-scale deployments, the high computational demand of complex algorithms can contribute to increased energy consumption and carbon footprint. This poses ethical and environmental concerns.

Project Idea

- Our project aims to develop a novel encryption algorithm based on polynomial roots, focusing on achieving both high speed and strong security. The goal is to surpass existing encryption algorithms, such as AES, in terms of speed. Additionally, we plan to create a web application that utilizes our algorithm to securely and swiftly transmit messages over networks by encrypting and decrypting them.

➢ Root Finding Algorithms

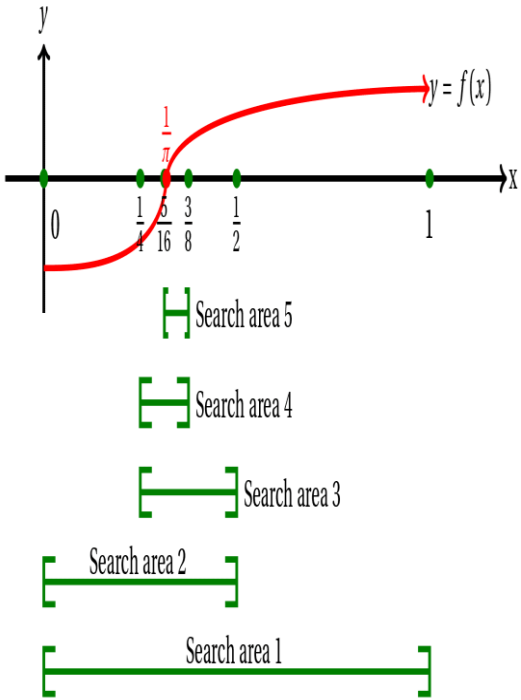
Before talking about our solution ,let's talk about the base of our algorithm which is root finding algorithms. Root finding algorithms are used in many applications, like machine learning to solve optimization problems, in computer graphics, in economics and finance, and in cryptography. Here we use the root finding algorithms in cryptography.

Bisection Method

One of the most popular root finding algorithms is the bisection method.

Bisection method works by dividing the interval in half and then determining which subinterval contains the root. This process is repeated until the root is found with the desired accuracy.

It calculates the midpoint $\frac{a+b}{2}$ of the interval and then checks if the midpoint is the root or if the root is in the left or right subinterval.



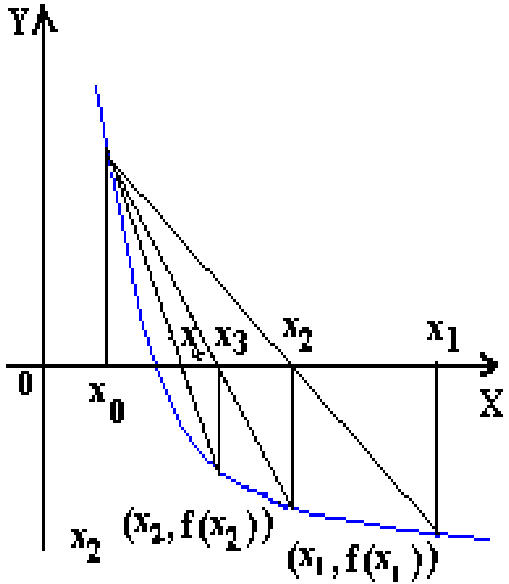
False Position Method

Another root finding algorithm is the false position method.

The false position method is similar to the bisection method, but instead of dividing the interval in half, it use a linear interpolation to determine the next approximation of the root.

The successive estimates, from the following relationship:

$$x = a - \frac{f(a).(b-a)}{f(b)-f(a)}$$



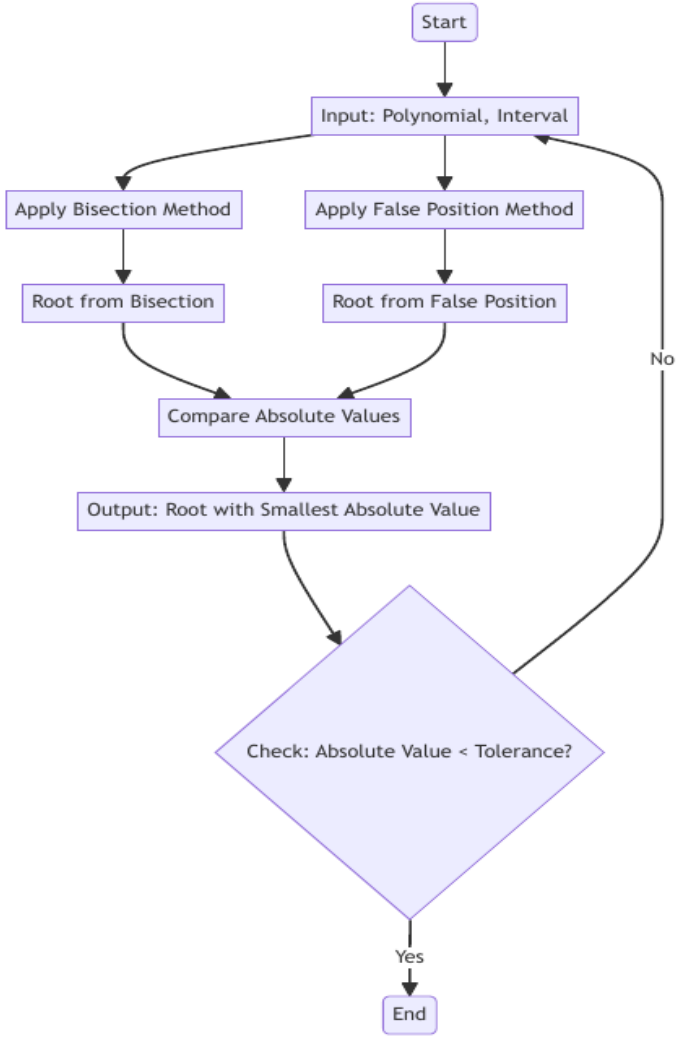
HybridBF Algorithm

The HybridBF algorithm is a root finding algorithm that combines the bisection method and the false position method to improve the convergence speed and accuracy of root finding.

We have tested this algorithm against both bisection and false position and it has shown to be faster and more accurate.

It Works by using the bisection and false position methods in parallel and then selecting the best approximation of the root from the two methods.

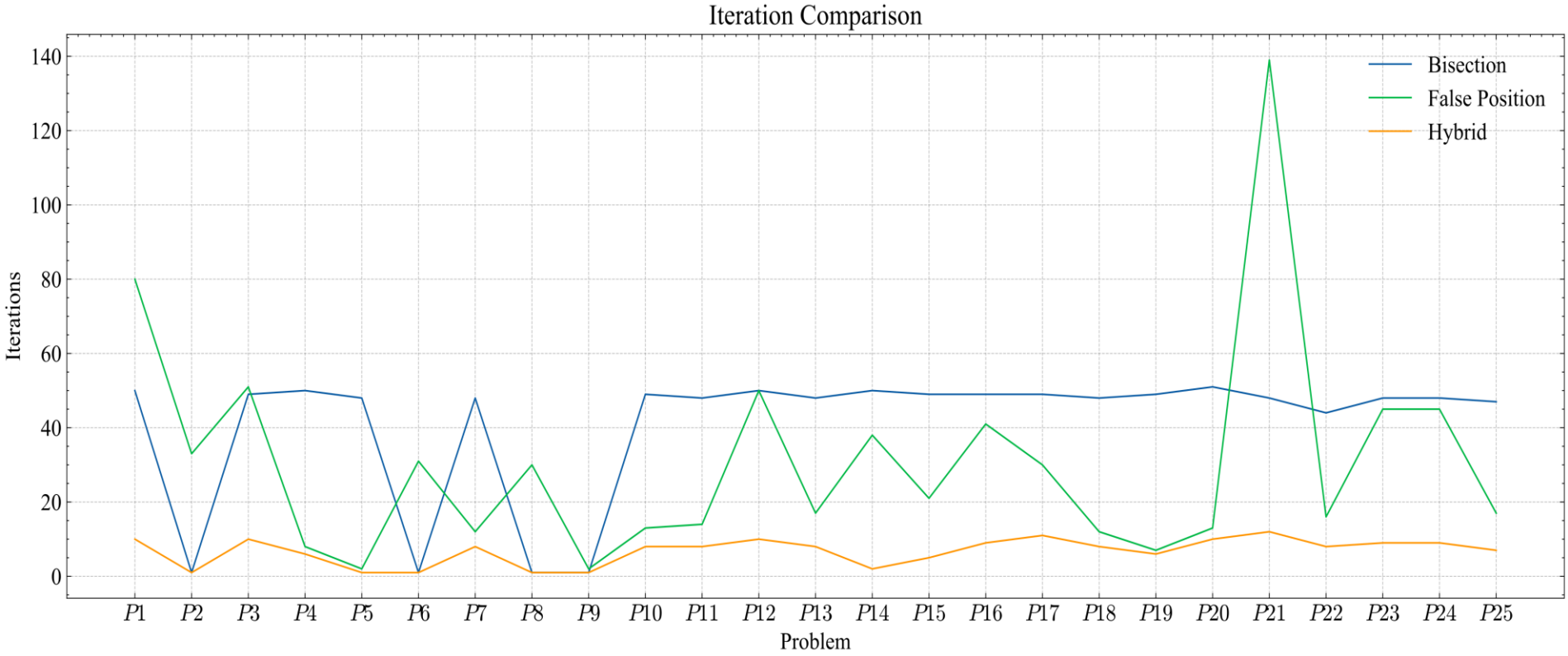
Hybrid Algorithm Flowchart



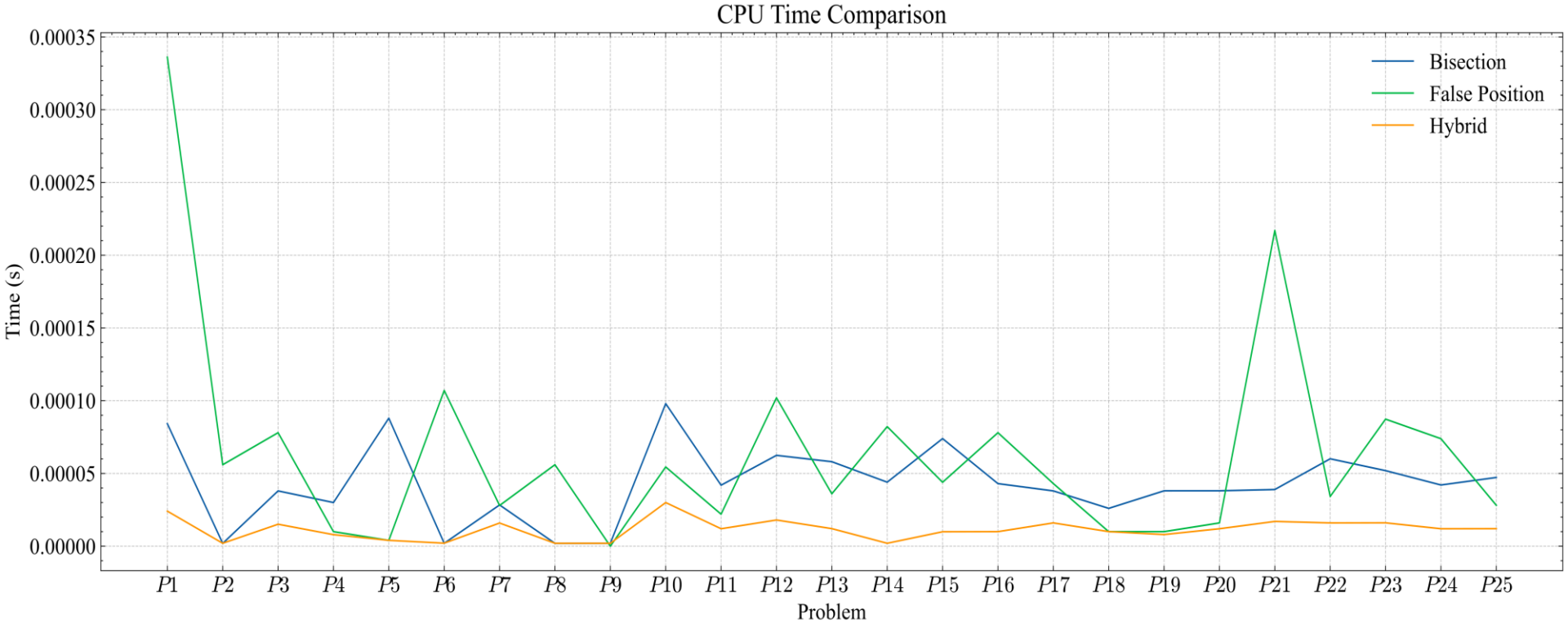
Compare Results

- We have compared the performance of the HybridBF algorithm with the bisection and false position methods using a set of benchmark functions and the results show that the HybridBF algorithm outperforms both bisection and false position in terms of convergence speed and time, number of iterations, and accuracy.

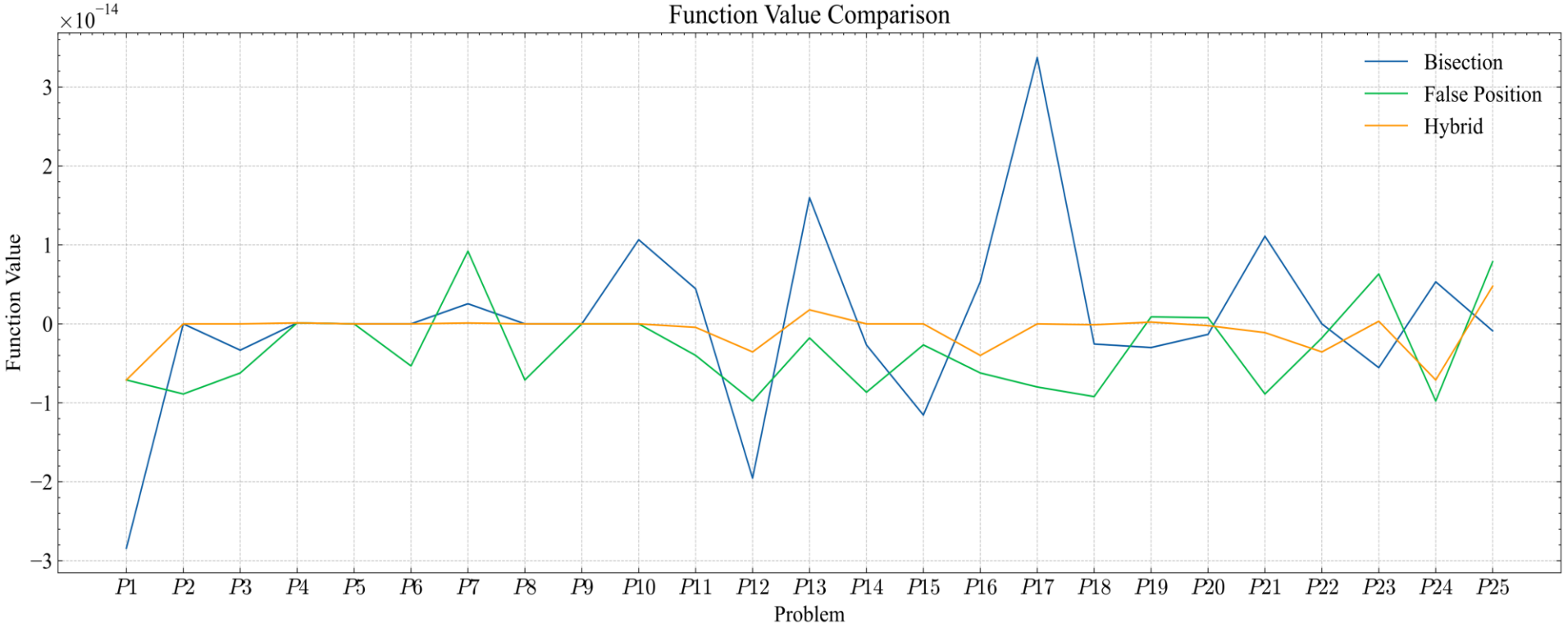
Number of Iterations



CPU Time


CPU Time Comparison
Bisection
False Position
Hybrid
Time (s)
0.00035
0.00030
0.00025
0.00020
0.00015
0.00010
0.00005
0.00000
P1
P2
P3
P4
P5
P6
P7
P8
P9
P10
P11
P12
P13
P14
P15
P16
P17
P18
P19
P20
P21
P22
P23
P24
P25
Problem

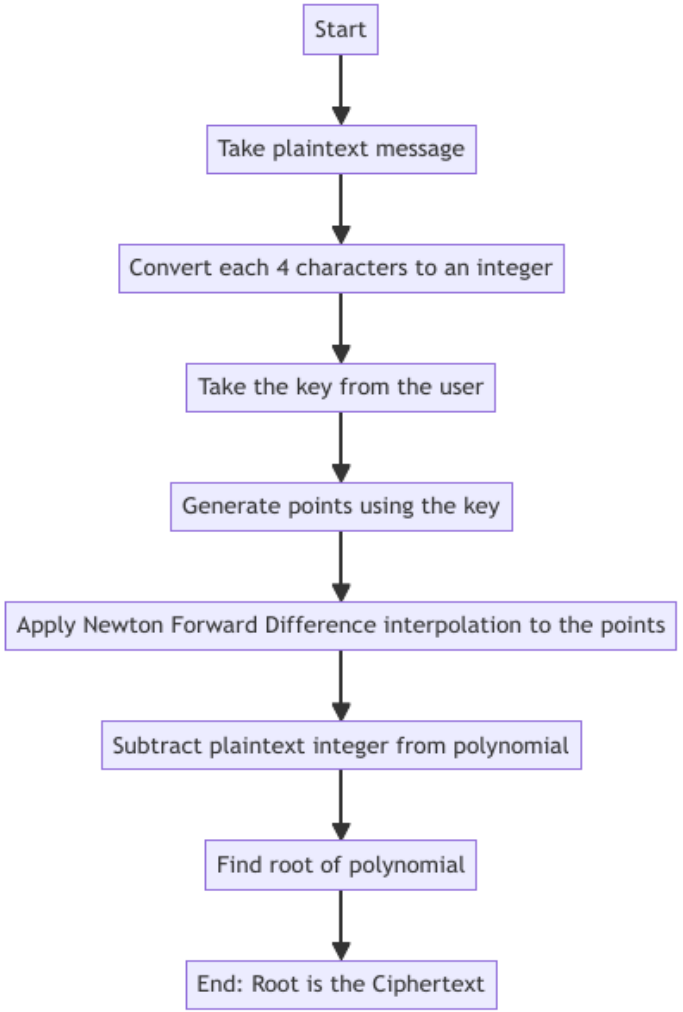
Function Value



➢ How Does Our Encryption Algorithm Works

- The algorithm is based on the roots of a polynomial. The roots of a polynomial are the values of that make the polynomial equal to zero. The algorithm uses these roots to perform encryption and decryption operations. The algorithm accepts a plaintext message and a key as input that will be used to generate a polynomial and produces a ciphertext message as output. The ciphertext is the root of that polynomial that can be decrypted to recover the original plaintext message.

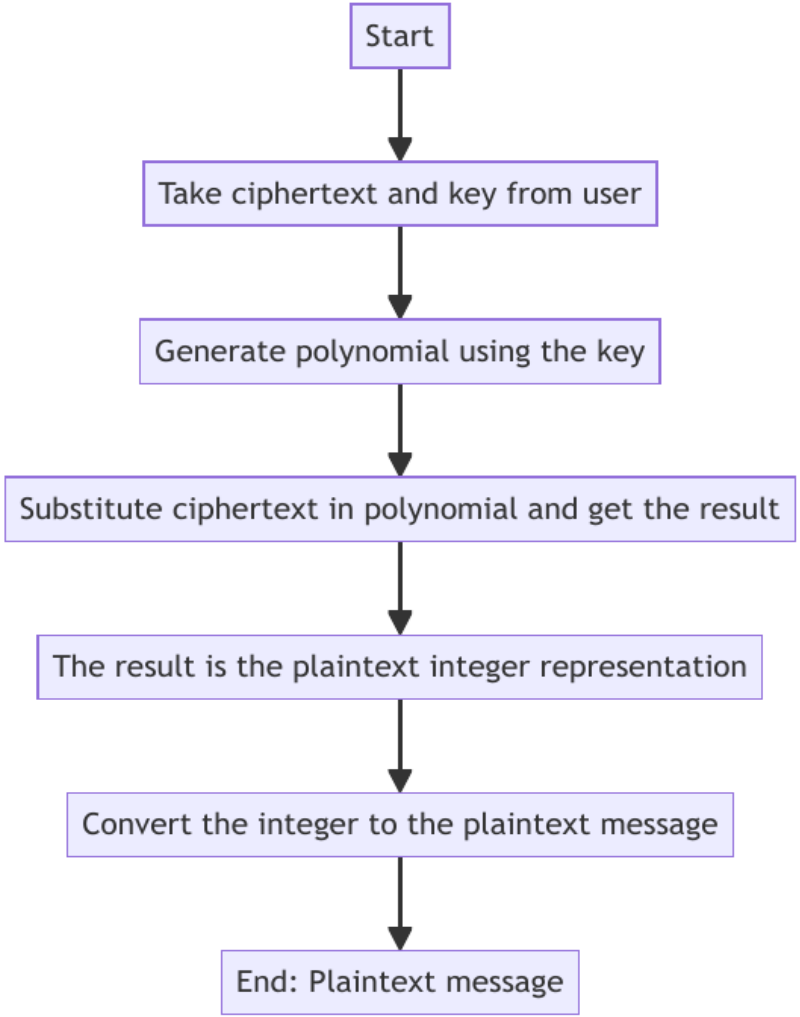
Encryption Process



Key Numbers

- x^+ End of x interval
- x^- Start of x interval
- y Start of y interval and the end will be the same value but negative to ensure there is a root
- s Number of Sections
- r Random State

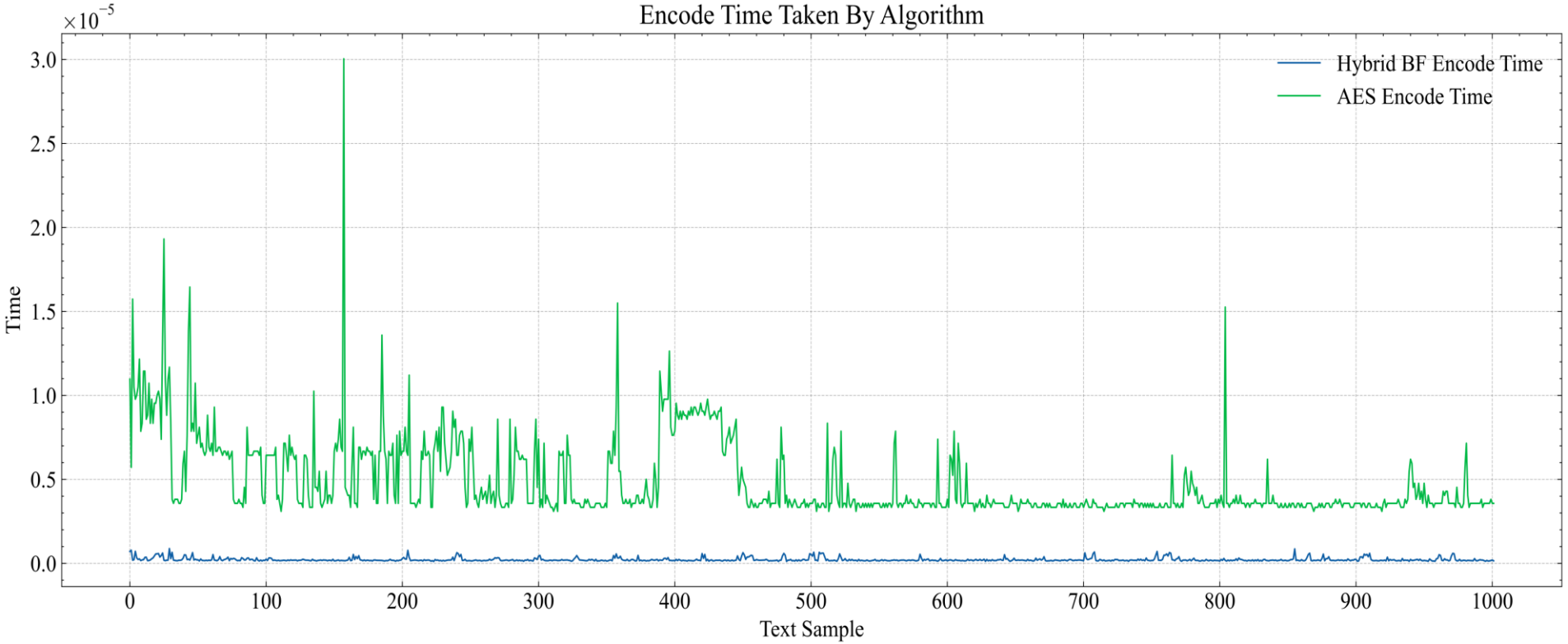
Decryption Process



Results

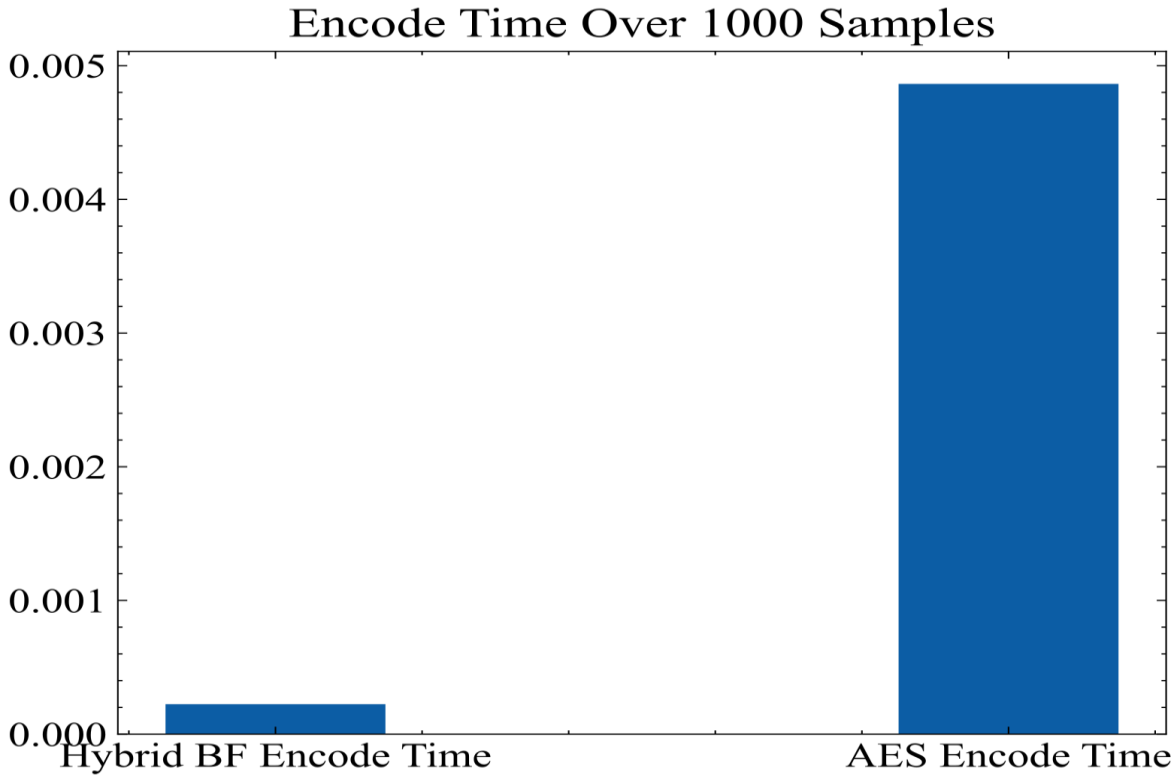
- The algorithm was tested using different plaintext messages and keys and was compared against AES encryption algorithm which is a symmetric encryption algorithm. The results showed that the algorithm is much faster than AES.

Encode Time Over 1000 Messages

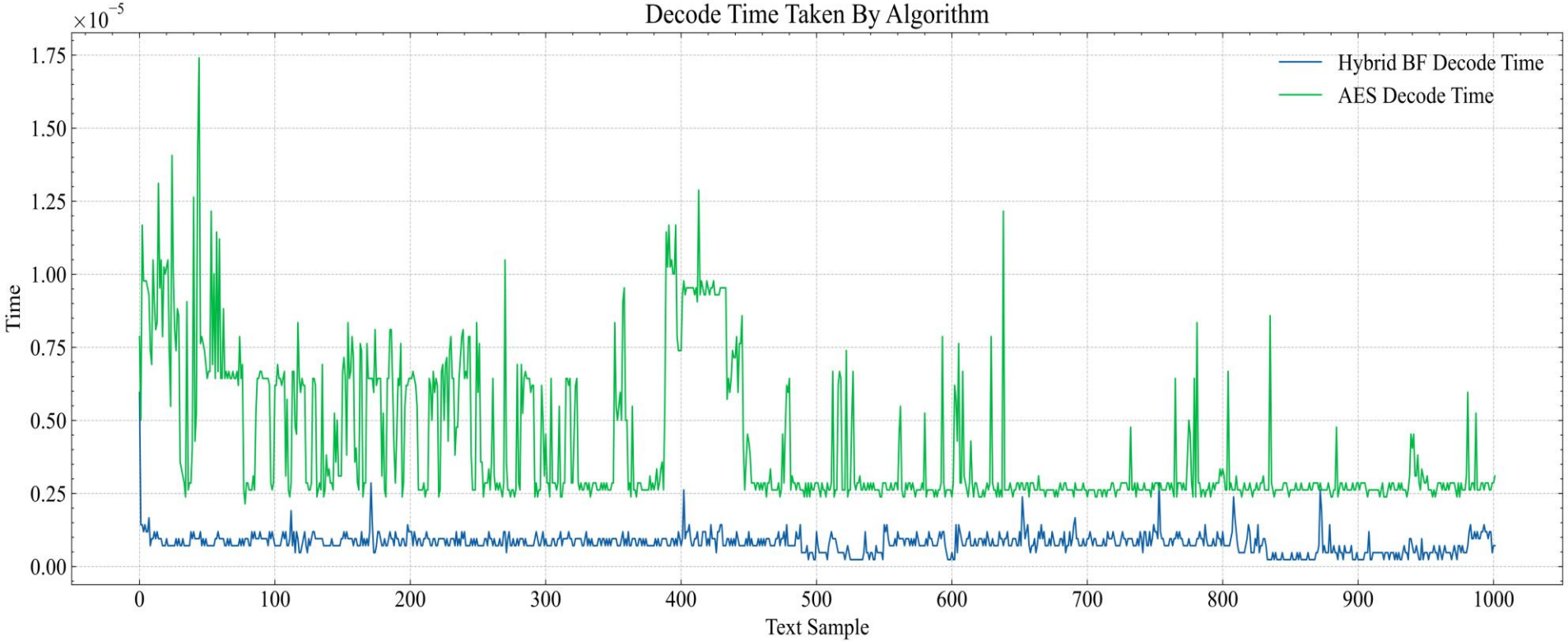


Encode Time Sum Over 1000 Messages

HybridBF broved to be 20 faster in total time

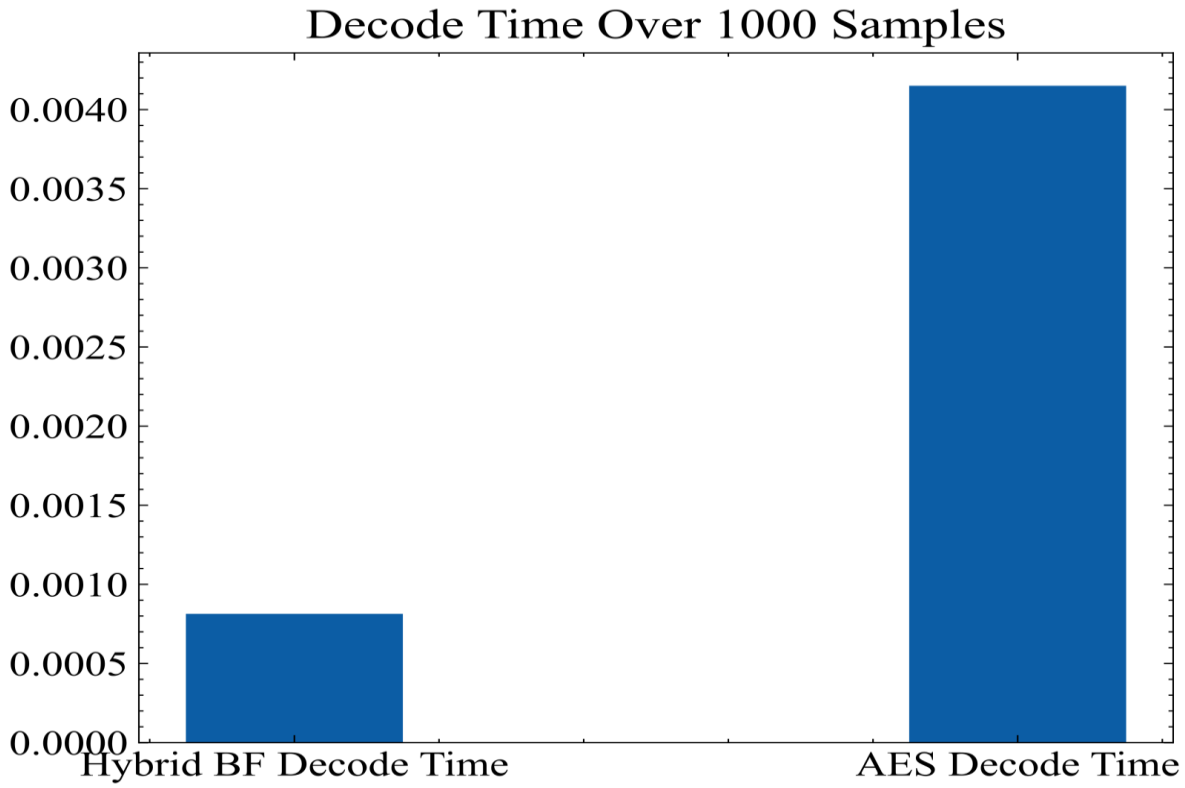


Decode Time Over 1000 Messages



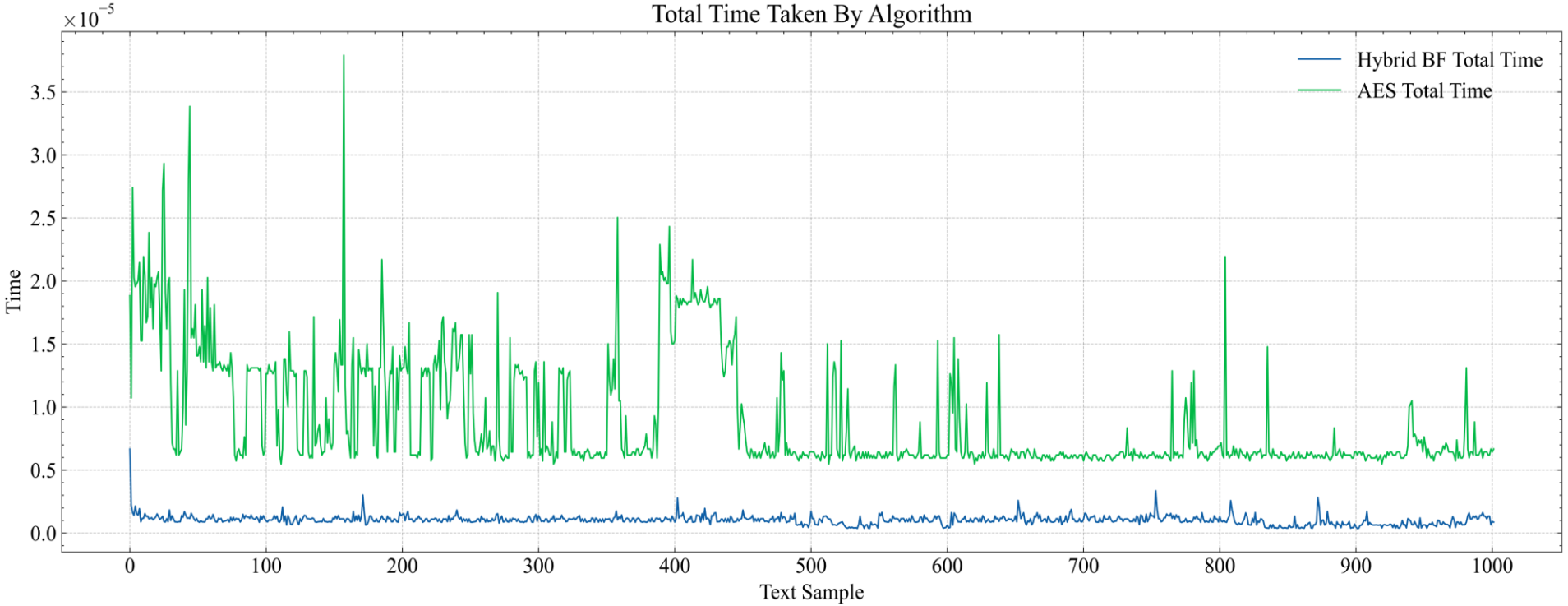
Decode Time Sum Over 1000 Messages

HybridBF broved to be
8 faster in decode time



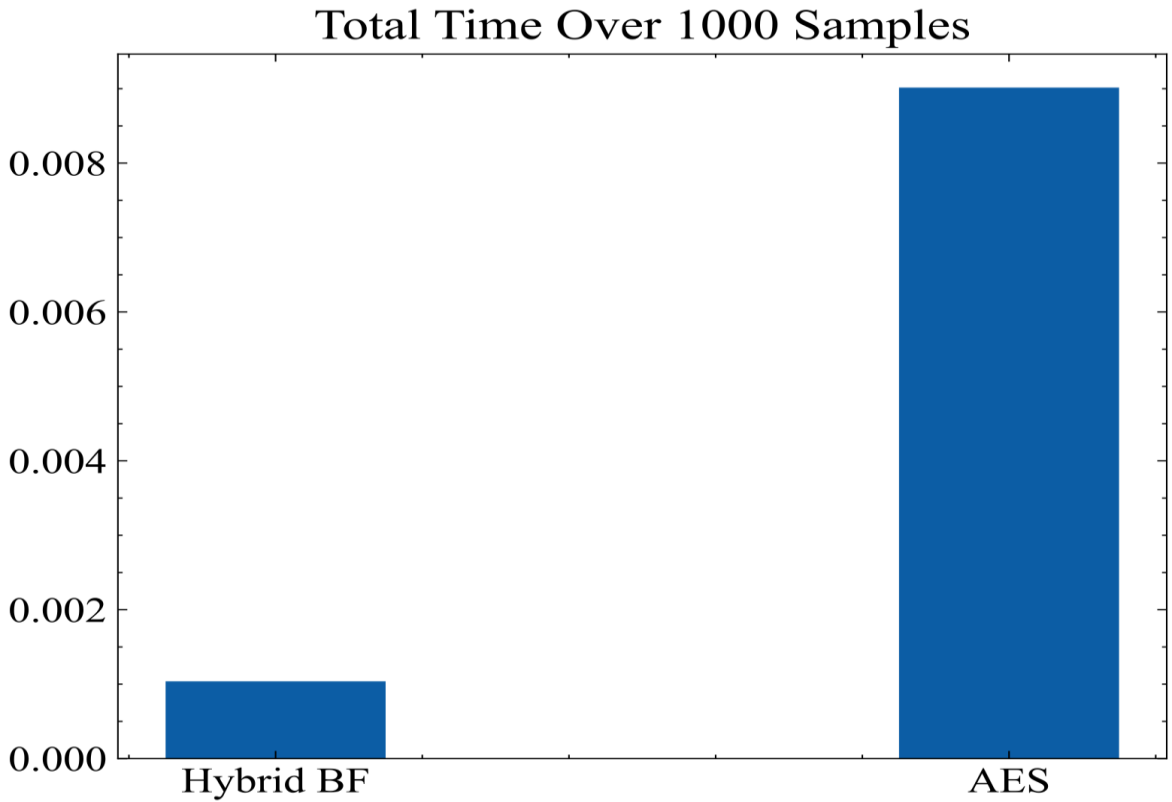
Total Time Over 1000 Messages

- We mean by total time the sum of the encode time and the decode time for each message.



Total Time Sum Over 1000 Messages

HybridBF broved to be 9 faster in total time



➢ Advantages of Faster Encryption Algorithms

- Enable real-time encryption of data streams, like encrypting voice calls or video chats.
- More widespread adoption and use of encryption.
- Make full disk encryption more practical.
- Shift cryptography workloads to the client side.

Current Solutions in the Market

There are well-established encryption algorithms available, both symmetric and asymmetric, that have been extensively researched, tested, and proven secure.

- Symmetric encryption algorithms like AES.
- Asymmetric encryption algorithms like RSA and ECC provide secure key exchange.
- Existing solutions also include cryptographic libraries and frameworks that provide implementations of various encryption algorithms.

➢ Methodology

Tools & Libraries:

- Python
- Pandas
- Matplotlib
- Numpy
- Scienceplots
- Scipy
- Pycryptodome
- Time
- String

Current Progress

Our encryption algorithm has been successfully implemented and tested using various messages. Through rigorous comparisons with existing encryption algorithms such as AES, our algorithm has demonstrated significant improvements in terms of speed. Additionally, we have created diverse diagrams showcasing the functionality and features of our messaging web application.

Future Work

During the second phase of our project, we will focus on developing a messaging web application that leverages our high-speed encryption algorithm to ensure secure communication between users. We will also implement a robust approach to encrypting passwords using a secure asymmetric encryption algorithm like RSA within the application's development process.

THANKS!