Shape, logo

Description automatically generated

Department of Computer Science

CS 528 – Data Privacy and Security

Project Report

## Secure Multi-Party Auction Platform

By

Rik Ganguli Biswas (A20513718)

Pradyumna Deshpande (A20517693)

Shekhar Kausal (A20512180)

### Contents

1. [Team Description 3](#_Toc165669405)
2. [Introduction 4](#_Toc165669406)
3. [Implementation 6](#_Toc165669407)
4. [Results 10](#_Toc165669408)
5. [Scope for Improvement 12](#_Toc165669409)

# 1.Team Description

The project team consists of three members: Rik, Pradyumna, and Shekhar. They are currently in their final semester pursuing Master of Science in Computer Science (MCS). Each member has a background in Computer Science, having completed their undergraduate education in the same field. After completing their undergraduate studies, Pradyumna and Shekhar gained experience in the private sector, while Rik continued his academic pursuits. All team members have previously completed the Introduction to Information Security (CS-458) course, which ignited their interest in Cybersecurity. Consequently, they decided to further explore this field by opting for the Data Privacy and Security (CS-528) course.

# 2.Introduction

Encryption techniques come in many shapes and forms. There are countless algorithms starting from Caesar ciphers to AES. Most of the advanced algorithms use a variety of techniques in combination to achieve highly complex and hard-to-break algorithms. In this project, we have used Secure Multi Party computation using Additive Modulus Algorithm and RSA Algorithm to encipher/decipher the sensitive fields. We have also used Yao’s Millionaires’ Protocol to perform mathematical calculations on the enciphered data.

Below is the breakdown of the concepts that have been used in the project –

### A. Additive Modulus Algorithm

Additive modulo encryption is an algorithm that uses module addition to encrypt and decrypt data by shifting each letter of the plaintext alphabet to create a ciphertext alphabet. In decryption, you subtract instead of add. The modulo operator, which is often represented as '%', is a fundamental mathematical operation that calculates the remainder when one number is divided.

### B. RSA Algorithm

**RSA** (**Rivest–Shamir–Adleman**) is a public-key cryptosystem, one of the oldest widely used for secure data transmission. The initialism "RSA" comes from the surnames of Ron Rivest, Adi Shamir, and Leonard Adleman, who publicly described the algorithm in 1977.

In a public-key cryptosystem, the encryption key is public and distinct from the decryption key, which is kept secret (private). An RSA user creates and publishes a public key based on two large prime numbers, along with an auxiliary value. The prime numbers are kept secret. Messages can be encrypted by anyone, via the public key, but can only be decrypted by someone who knows the private key.

The security of RSA relies on the practical difficulty of factoring the product of two large prime numbers, the "factoring problem". Breaking RSA encryption is known as the RSA problem. Whether it is as difficult as the factoring problem is an open question. There are no published methods to defeat the system if a large enough key is used.

### C. Yao’s Millionaires’ Protocol

**Yao's Millionaires' protocol** is a secure multi-party computation problem introduced in 1982 by computer scientist and computational theorist Andrew Yao. The problem discusses two millionaires, Alice, and Bob, who are interested in knowing which of them is richer without revealing their actual wealth. This problem is analogous to a more general problem where there are two numbers ‘a’ and ‘b’ and the goal is to determine whether the inequality a ≥ b is true or false without revealing the actual values of ‘a’ and ‘b’.

# 3. Implementation

Our implementation is based on **Secure Multi Party Computation** where the user only knows his input and the final output. He is not able to view other users’ data. We have implemented this using RSA encryption. We have also implemented **Homomorphic encryption** more specifically, **Yao’s Millionaires’ Protocol** where we encrypt the values using **Additive Modulus Encryption** and the perform > operation to find the maximum value from the bids placed in the auction. We are using ‘Online Auctions Dataset’ from Kaggle (<https://www.kaggle.com/datasets/onlineauctions/online-auctions-dataset>). Below is the detailed implementation.

Our algorithm operates in two stages. In the initial stage, we utilize the dataset 'auctions.csv' following preprocessing and cleaning of the data. We apply Additive Modulus Encryption to the 'bid' column and RSA encryption to the 'bidder' column. Additionally, we validate that the bid\_time falls within the range of 0 to bid\_end time and that the placed\_bid is greater than or equal to the opening\_bid. By imposing these constraints, we exclusively consider bids that meet these criteria. Subsequently, we store the encrypted data, including the bidder's username, bid amount, product name, and auction\_id, in an empty dataframe.

A screen shot of a computer program

Description automatically generated

Fig.1 – Code Snippet for Additive Modulus

A computer screen shot of a program code

Description automatically generated

Fig.2.a – Code Snippet for RSA Encryption

A computer screen shot of a program code

Description automatically generated

Fig.2.b – Code Snippet for RSA Encryption

A computer screen shot of a program code

Description automatically generated

Fig.2.c – Code Snippet for RSA Encryption

In the second stage of our algorithm, we employ Yao's Millionaires' Protocol to determine the maximum bid among all encrypted bid values. In the initial stage, we employed Additive Modulus encryption to facilitate the subsequent application of the Millionaires' Protocol. This allowed us to efficiently compare all bids and determine the maximum bid among them. Subsequently, we compile a new dataset named 'winners.csv', which consolidates data on the maximum bid, along with the bidder's username, auction\_id, and product name. This dataset serves as a resource for users to ascertain whether they were successful in the auction.

A screenshot of a computer program

Description automatically generated

Fig.3 – Code Snippet for Millionaires’ Protocol

Users utilize their private key to decrypt the dataset, enabling them to ascertain their auction outcomes. They are restricted to viewing only their own outcomes, as any data pertaining to other users remains encrypted from their view. For instance, when a user employs their private key to access the dataset and observes their username, it indicates that they have emerged as the winning bidder for a product in the auction. Conversely, if a user fails to secure a bid, they will encounter an improperly decrypted text in place of the username. This signifies that no other user, apart from the winner, can discern who secured the bid. Please refer attached picture in Results.

# 4. Results

A screen shot of a computer

Description automatically generated

Fig.4 – Code Snippet for final output

A screenshot of a computer

Description automatically generated

Fig.5 – Code Snippet for auctions.csv

A screenshot of a computer

Description automatically generated

Fig.6 – Code Snippet for winners.csv

# 5. Scope for Improvement

Presently, our RSA encryption implementation relies on smaller prime numbers due to the significant computational burden posed by larger primes. Despite incorporating code to accommodate larger primes, we've opted for smaller ones to mitigate computational overheads. Looking ahead, as computational capacities advance, we anticipate transitioning to larger primes and generating unique key pairs for each user.

Currently, our output is limited to the console display. However, improving user interaction with a simplified interface shows potential for expanding functionality. This interface could facilitate the integration of extra features. A web-based application stands out as the optimal platform for developing a comprehensive auction platform.