Project Report – Gin Rummy with Advanced Technologies

---

1. Background and Introduction

Project Background

Gin Rummy is a classic card game that combines strategy, luck, and mathematical calculation. We chose this project because it allows us to integrate various development areas – real-time application development, user interface design, and the implementation of advanced cryptographic solutions. Additionally, this project provided an opportunity to deepen our knowledge of modern technologies such as Flask, React, ElGamal encryption, and Zero Knowledge Proof (ZKP).

Project Objectives

Develop a network-based Gin Rummy game:

- **Build a server in Python** using Flask and Socket.IO to manage game logic and real-time communication.
- **Develop a dynamic front-end** using React to provide an interactive user interface.
- **Implement cryptographic solutions:**
    - **ElGamal Card Encryption:** Ensure that cards are encrypted so that only authorized players can decrypt them.
    - **Secure Deck Shuffling with ZKP:** Develop a ZKP module to validate the shuffling process in a secure manner – each party shuffles the deck 100 times and verifies the process without revealing the full order.
    - **Discard Validation using ZKP:** Create a mechanism where, when a player discards a card, they prove (without revealing their entire hand) that the discarded card was indeed part of their hand.

---

2. Technologies Used

Core Technologies in the Project

Flask and Socket.IO

- **Flask:** A lightweight Python framework used to build the server and create an API for managing the game logic.
- **Socket.IO:** Enables real-time communication between the server and clients, ensuring that every game action (such as drawing, discarding, or rearranging cards) is updated immediately.

React

A JavaScript library for building dynamic user interfaces that allow real-time updates based on data received from the server.

ElGamal Encryption

A cryptographic algorithm used to encrypt card values using public and private keys, ensuring that only players with the proper keys can decrypt the cards.

Zero Knowledge Proof (ZKP)

We implemented ZKP in two key areas:

- **Secure Deck Shuffling:**
    - A module was developed to securely validate the shuffling of the deck, where each party shuffles the deck 100 times and validates the process via cryptographic commitments based on a hash function.
- **Discard Validation:**
    - A mechanism was created in which, when a player discards a card, they prove (without revealing their entire hand) that the discarded card was part of their hand. This is achieved by generating commitments for every card in their hand at the start of the turn and then producing a discard proof when the card is thrown away.

3. Challenges and Solutions

Technological Challenges

Managing and Encrypting Cards

- **Challenge:** Develop a system that encrypts the card values so that only the players can decrypt them.
- **Solution:** Implement the ElGamal algorithm to generate public and private keys, enabling encryption and decryption through cooperative use of two private keys.

Secure Deck Shuffling with ZKP

- **Challenge:** Ensure that the deck shuffling is conducted fairly by both parties without revealing the complete card order.
- **Solution:** Develop a ZKP module that simulates secure shuffling—each party (Alice and Bob) shuffles the deck 100 times and verifies the process using cryptographic commitments based on a hash function.

Discard Validation

- **Challenge:** Ensure that the card discarded was indeed part of the player's hand, without revealing the rest of the hand.
- **Solution:** At the beginning of a turn, each player creates cryptographic commitments (using a hash function with a random salt) for every card in their hand. When discarding, the player provides a ZKP (discard proof) that validates the discarded card against the initial commitments. The server then recomputes the hash and checks for a match to confirm the discard's validity.

Real-Time Communication and Synchronization

- **Challenge:** Ensure that all updates and actions within the game occur in real time and are synchronized across all clients.

- **Solution:** Utilize Socket.IO to handle real-time communication, ensuring that every action (card draw, discard, knock, etc.) is instantly propagated to all players.

Design Challenges

User Interface Design

- **Challenge:** Create an intuitive, attractive, and user-friendly interface incorporating well-designed graphical elements (such as stylized cards, buttons, and headings).
- **Solution:** Use React to build dynamic components and define custom styles using CSS/SCSS that match the desired color schemes (e.g., burgundy, white) and typography.

Data Display and Game State Updates

- **Challenge:** Clearly display the game state – such as the number of cards remaining, the current player's turn, and scores – in real time.
- **Solution:** Develop a game board component that uses Socket.IO to display and update this information interactively.

---

4. Results and Achievements

Key Achievements

- **Real-Time Game System:**
    - A Flask server was built to manage game logic, including card distribution and turn management, with real-time updates via Socket.IO.
- **ElGamal Card Encryption:**
    - Cards are encrypted using the ElGamal algorithm, ensuring that only players with the appropriate keys can decrypt the card values, thereby providing a high level of security.

- **Secure Deck Shuffling with ZKP:**
    - A ZKP module was developed that simulates a secure deck shuffling process where each party shuffles the deck 100 times and verifies the process using cryptographic commitments, thus ensuring a fair shuffle without revealing the complete order.
- **Discard Validation using ZKP:**
    - A mechanism was added that allows a player to prove, without revealing their entire hand, that the card they discard was indeed part of their hand. This is achieved by creating commitments for every card at the start of the turn and generating a discard proof when the card is thrown.
- **Interactive User Interface:**
    - The React-based front end displays the game dynamically with real-time updates, enabling each player to follow the game state and perform actions interactively.

Evaluation of Achievements

The project successfully meets its defined objectives by combining advanced technologies from real-time development and cryptography. The implemented solutions (ElGamal encryption and ZKP for both deck shuffling and discard validation) provide a secure and fair game environment while delivering an interactive user experience.

---

5. Conclusion and Future Recommendations

Conclusion

Project Objective:

Develop a network-based Gin Rummy game that integrates advanced technologies – Flask, React, ElGamal encryption, and ZKP – to ensure data security and fair gameplay.

Achievements:

- A real-time game system with immediate updates.
- Card encryption using the ElGamal algorithm.
- Secure deck shuffling and discard validation using ZKP, ensuring that actions are executed lawfully and securely.
- An interactive and user-friendly interface for players.

Challenges:

Addressing technical challenges in cryptographic implementation and real-time synchronization, as well as designing an intuitive interface.

Future Recommendations

Enhance Communication Security:

Beyond using HTTPS/WSS, additional authentication layers can be added to Socket.IO messages.

Develop Comprehensive Testing:

Implement an extensive suite of unit and integration tests to verify the integrity of all system components, including the cryptographic modules.

Further UI Improvements:

Continue refining the user interface by adding animations and richer interactions to enhance the overall user experience.

Expand ZKP Implementation:

While the current implementation covers secure deck shuffling and discard validation, future work could explore applying ZKP to other aspects of the game as needed.