Poker on the Internet

Timothy Fong
Computer Engineering Department
San Jose State University
San Jose, CA USA
timothy.fong@sjsu.edu

Haely Shah
Electrical Engineering Department
San Jose State University
San Jose, CA USA
haely.shah@sjsu.edu

Oliver Zhu
Computer Engineering Department
San Joe State University
San Jose, CA, USA
oliver.zhu@sjsu.edu

Abstract
This paper reports on the design, implementation, and evaluation of a distributed Internet-based poker service using encryption schemes such as Rivest Shamir Adleman (RSA) and Advanced Encryption Standard (AES) to ensure fairness and reliability through authentication, access control, confidentiality, and nonrepudiation. In this paper, we describe our implementation of Poker on the Internet (POTI) and describe in detail all the features we implemented to protect against common attacks by attackers and malicious players while maintaining all the common aspects of a poker game. This project demonstrates how to establish modern security system as well as the functionality of system described.

Keywords
Poker, Encryption, RSA, AES, signature validation

1. Introduction

Poker is an old game that combines the exchange of money, strategies with incomplete information, random luck, and the intricate coordination between the players and a neutral dealer. Games, however, are been growing in complexity ever since we entered the digital age and changes in games range from the development of new video games to the digitization of older games. However, the high stakes nature of poker and its fundamental information constraints demand a playing environment that provides fairness and reliability and despite all the changes of the digital age, some aspects of games do not change. One unfortunate constant phenomenon that continues is the cheating. A review of current literature has revealed that, though the concept of cheating has not changed, the means by which cheating is performed has changed drastically to adapt to modern day platforms [1]. The term “hacking” which is defined as “unauthorized access to or control over computer network security systems for some illicit purpose” is now considered a variant of cheating. In the case of our poker project, hacking can be used for direct modification of game values to affect/disrupt gameplay or illegal access to the database to modify server-restricted data. In either case, necessary precautions need to be taken to prevent cheaters from prevailing. The purpose of this report is to present our results from developing a POTI system and demonstrate its features as a fair and stable game program. We will explore our POTI application to validate the need for security and authentication to prevent malicious individuals from tampering with the game. We used publicly

available security algorithms: RSA and AES cryptosystems for secure data transmission because both systems have been publicly acknowledged to be nearly impossible to break given a large enough key size. Data signatures are employed to authenticate messages sent from the client to the server. All keys for a client will be destroyed at the end of a session to prevent access to an old session. Finally, the database for storing all relevant user information and keys is secured any accessible only by the server. In section 2, this report provides an overview on how to create an account for gameplay, the poker house-player interaction. In section 3, the results obtained from the application incorporating the system are displayed followed by an analysis in section 4 of the strengths and weaknesses of our system. Finally, in section 5, we ended with a conclusion and possible future developments

2. Methodology

2.1 Account Creation:

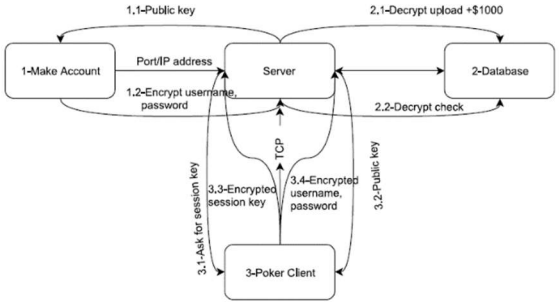

Figure 1 Account Creation Flowchart.

Prior to playing the poker application, players first need to create an account that will hold their username, password, and wallet information. Players would connect to the multithreaded server using a unique client and request a Public Key. The Private/Public Keys were generated using PyCrypto, a free Python library that provides secure hash functions and various encryption algorithms. We used the cryptographic libraries to implement the RSA algorithm with a key size of 1024 bits and a large random number to generate a Private Key from which we derived a Public Key. Then, they will encrypt their requested username and password using the aforementioned Public Key and transmit them to the server. The server will then decrypt and store the information on our MySQL database along with a starting amount of $1000. Finally, the unique client and the server thread will both close. Afterwards, whenever a player opens the poker application and sends the server the login information, the server can query the database to confirm authenticity and decide whether or not enter the player into the game session.

2.2 POTI Application

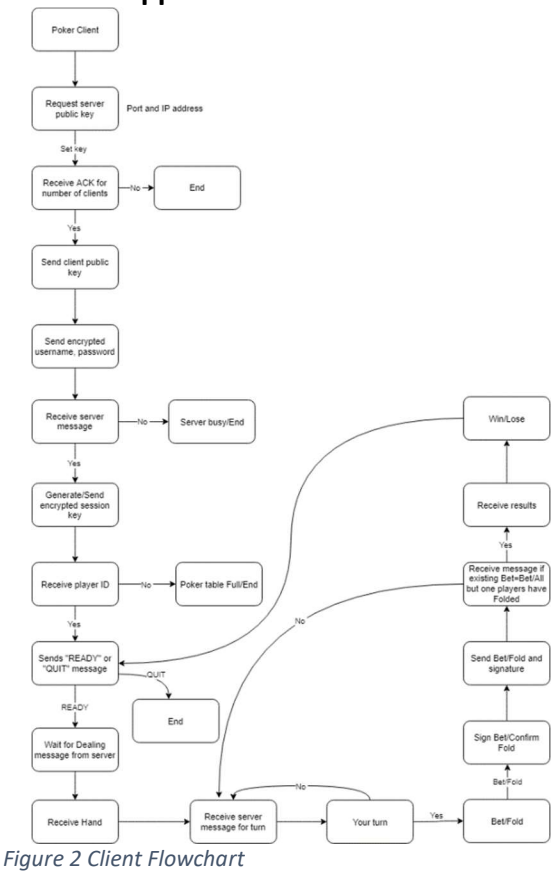

Figure 2 Client Flowchart

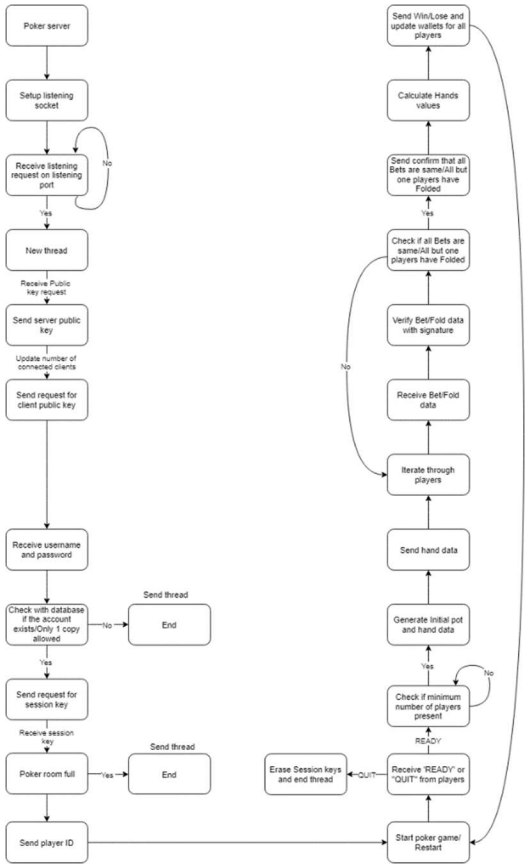

Figure 3 Server Flowchart

When the server initializes, it sets up a listening socket and waits to receive connection request from the clients. Once it receives the request, a new thread is created to handle the client. The server then receives requests for its Public Key and it sends in the key to its connected clients. The server then requests the clients' Public Keys. The clients then input their usernames and passwords, encrypts it with the server's Public Key, sends it to the server. The server decrypts the login information and checks with the database if the corresponding account exists or not. The thread closes if the account does not exist and if it does exist, then the server sends a request for the Session Key. After receiving the Session Key, the server updates the database with the Session Key and checks if the current number of players. If there are more than five players then the game session is considered full and the server thread and client are closed. Otherwise, the server gives the client a ID number from a list. The server then waits to receive a READY or a QUIT prompt from the clients. If the server receives a QUIT prompt instead, the client's Session Key is erased from the database and the thread ends. If the server receives a READY prompt, it puts the client in a waiting queue until a minimum of two players are in the session. After the minimum players are reached, the server deals the cards and takes the initial ante of $10 from each player. The players now start playing and send Bets and bet signatures. The server verifies the Bets using the signature, the game goes on till all existing Bets are the same. The server then sends the confirmation message to the client that the betting stage is over and calculates hand values. The game results are sent to every player and the wallets are updated on the database. Finally, the server waits for the READY or QUIT prompts again to start another round.

2.3 Poker Logic

The cards are then randomly shuffled and dealt. Each hand is sorted in a decreasing order and a partial score is calculated for each of the players using the following equations:

Sorted Hand = [Card with the Highest Rank (R0), Card with the 2nd Highest Rank (R1), Card with the 3rd Highest Rank (R2), Card with the 4th Highest Rank (R3), Card with the Lowest Rank (R4)]

Partial Score = (R0 x 13^5) + (R1 x 13^4) + (R2 x 13^3)+ (R3 x 13^2) + R4

The partial score prefers a higher ranked card against the lower ranked ones, thus

helpful in breaking the tie when all the players have a High Card.
The Poker hand winning combinations are then ranked and provided a multiplying index (h). The Royal Flush would have the highest multiplying index (h=10,000 here), and the High Card would have the lowest multiplying index (h=100 here). This is helpful in calculating the total score thus deciding who wins the round.
The Total Score is given by:

$$Total\ Score = h \times 130^5 + Partial\ Score$$

The player with the maximum Total Score ultimately wins the round. In case of a Tie, the second highest card is taken into consideration.

3. Results

The system was tested with respect to functionality and security.

3.1 POTI System

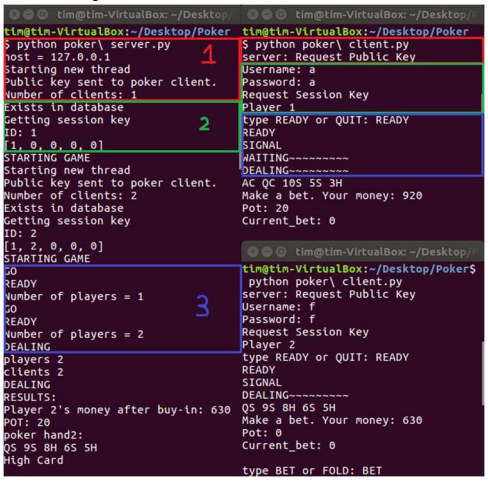

Figure 4 (1) Initial Connection. (2) Checking the database. (3) Joining the game.

In Figure 4 section 1 marked in red, the server receives a client and starts a thread to handle each of them. The client requests the Public Key from the server and upon receiving the key, the user has to input a username and password. In Figure 4 section 2 marked in green, the client sends the encrypted username and password to the server and the server decrypts and confirms with the database that the username and password exist on the database. Next, the server requests a session key from the client. The client generates a Session Key and sends it encrypted to the server. The server decrypts the Session Key and replies by giving each client a Player ID (Player 1, Player 2, etc.) and requests that they input "READY" to join the game. In Figure 4 section 3 marked in blue, the server confirms that a minimum of two players have readied and begins the round.

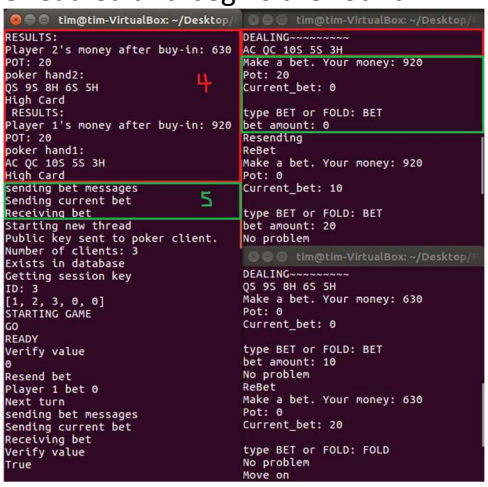

Figure 5 (4) Ante and cards. (5) First round of betting.

In Figure 5 section 4 marked in red, the server deals a hand of five cards to each player. The cards are named with a value and a suit. For example, the Ace of Clubs would be named 'AC'. In this case, Player 1 receives a hand with Ace of Clubs, Queen of Clubs, Ten of Spades, Five of Spades, and Three of Hearts. The hand is a "High Card" type of hand. In Figure 5 section 5 marked in green, the server receives the bet from the clients and verifies that the bet came from the client. Each bet is received in sequence, in this case Player 1 goes first and Player 2 goes last. All bets are then compared. Because the bet values are not equal, a second round of betting occurs

with the current bet set to the highest bet value in the previous round. In this case, Player 1 bet $0 but Player 2 bet $10. Therefore, a second round of betting will occur.

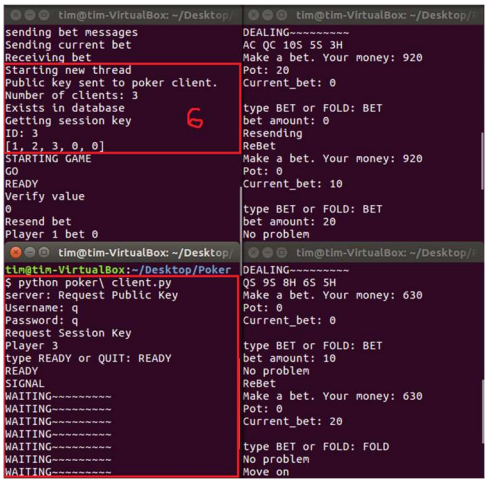

Figure 6 (6) Player in the waiting queue

Midway through the game, in Figure 6 section 6 marked in red, a third player tries to join the game. After the Player 3 finishes readying, the player is forced to wait until the current round is over.

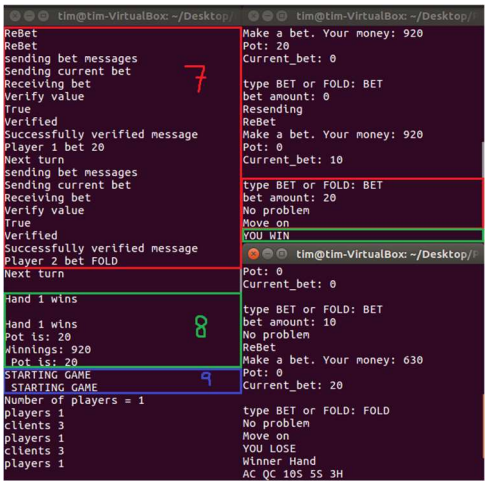

Figure 7 (7) Second round of betting. (8) Game Conclusion (9) Next round

In Figure 7 section 7 marked in red, the second round of betting occurs and continues until the bets are all equal or all other players fold. Here, we can see that after Player 1 re-raised, Player 2 folds. In Figure 7 section 8 marked in green, the round concludes with one of the clients winning and the database updates to include the amount won and the amount lost. The loser is given the privilege of viewing the winner's hand. In Figure 7, section 9 marked in blue, the game, pot, bets, and hands are reset and the player are required to ready up for another if they so choose. In this next round, because Player 3 had joined the waiting queue previously, the three players will be playing as seen in the section marked in red in Figure 8.

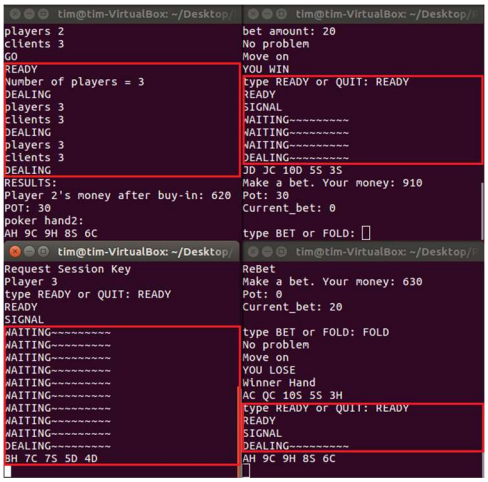

Figure 8 Player 3 joins the game.

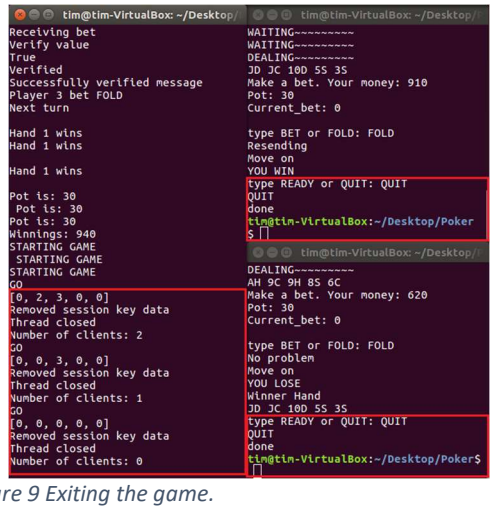

Figure 9 Exiting the game.

```
mysql> select * from users;
```

username	password	session_key	IV	Player_ID	Money
f	f	Ð®[illegible]È,%c,,¦[illegible]|d~	+¥–ÉC°Õ…T[illegible];”²	2	700
a	a	ÕÝM+þ?[illegible]–Œ[illegible]Ø”‘Y	T"0‹Û[illegible]o%-„1)ë¼f	1	950

2 rows in set (0.00 sec)

Figure 10 Poker database during a game.

If the player instead chooses to quit the game, their respective Player IDs are removed from the active list and the thread handling them are closed as seen in Figure 9.

3.2 MySQL Database

```
mysql> select * from users;
```

username	password	session_key	IV	Player_ID	Money
f	f	NULL	NULL	0	700
a	a	NULL	NULL	0	950

2 rows in set (0.00 sec)

Figure 11 Poker Database before and after a game.

As shown in Figure 11, before a game session, each account in the database will have no Session Key, Initialization Vector (IV), or Player ID.
However, in Figure 10, during a game session, the server will store the Session Keys, IV, and Player IDs of players currently active. After the exiting the game session, the server will erase the Session Keys, IV, and Player IDs and the database will revert to the state in Figure 11.

3.3 RSA and AES Encryption

Figure 12 (1) Server's Public RSA Key. (2) Client's Public RSA Key.

When starting the client and server, Public/Private Key pairs are generated each respective side and then shared when requested. At the beginning of the application, Public Keys are shared between client and server as shown in Figure 12 sections 1 and 2.
During the application, the Public/Private Key pairs are used to encrypt and decrypt important information, such as usernames, passwords and Session Keys as seen in sections 3 and 4 of the Figure 13. After the Session Keys are obtained by the server, all following information relating to the poker game are encrypted and decrypted using the Session Keys instead and the Public/Private Key pairs are used for signatures and verification.

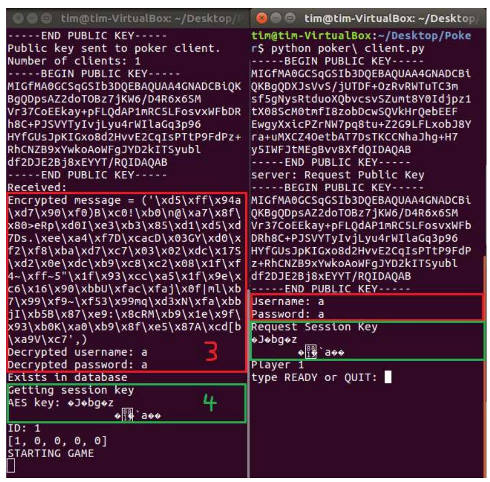

Figure 13 (3) Encrypted and Decrypted username and password. (4) AES Session Key.

3.4 Signature and Verification

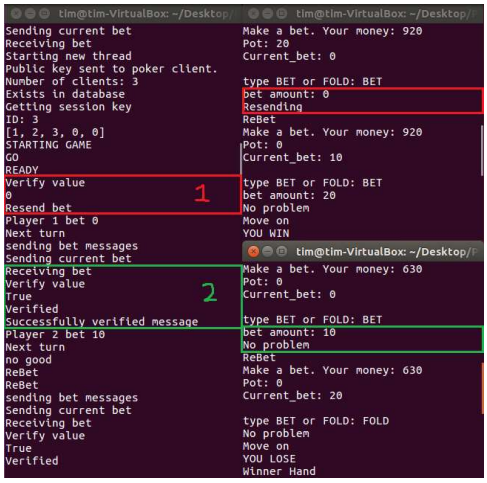

Figure 14 (1) Verify failed. (2) Verify confirmed.

After each bet is made, the bet is signed using the client's RSA Private Key and then sent together with the bet amount to the server. The server will decrypt the signature using the client's Public Key and verify with the bet whether or not the bet came from the client. If the verification was false, as seen in Figure 14 section 1 marked in red, the server will ask the client to resend the bet. However, if the verification was true, as seen in Figure 14 section 2 marked in green, the server will simply send a confirmation message to the client.

4. Discussion

4.1 Advantages

Casinos have been concerned with security issues since time immemorial. Constantly under threat by fraudsters, thieves, and scammers, casinos have been forced to come up with increasing security measures. However, for online casinos, these crimes can be difficult to detect, hard to trace, and almost impossible to prosecute. Faced with these challenges, we researched the most common security threats for online gaming and devised a number of ways for protecting the system against some of these risks.

Online games are prime targets for fraudsters for steal credentials and take over the existing accounts [2]. As a result, data privacy and confidentiality are not just an obligation but also a necessary measure to retain business. We have cryptosystems like RSA and AES to safeguard data confidentiality. User credentials are encrypted using the server's RSA Public Key for encryption and sent to the server and decrypted using the server's RSA Private Key.

The server must see to it that data sent over the network must not be changed mid-transit, and steps must be taken that data cannot be altered by unauthorized third parties [3]. We have implemented signatures for verifying that whether the Bet originated from a specific client and clients and the server make use of Session keys to send information or requests to one another. Session keys are destroyed after the game is over, thereby countering replay attacks and retaining the originality and freshness of the messages. When a potential player first signs in to access the Game, the server cross-references the credentials with the existing accounts in the database [4]. This check counters non-repudiation attacks and confirms the identities of the players.

We used MySQL to store the user credentials and wallets. MySQL has an inherent access privilege system protects against phishing attacks and unauthorized access and encrypts connections to the database using the TLS protocol. Finally, to further strengthen the system against random attacks, we have added salt to all the data packets that are being transmitted.

4.2 Disadvantages

We are aware that any vulnerability that is "too complicated for anyone to ever find" will be found. An online system that runs real time gaming can never be secure enough to not attract hackers or scammers. While our system is robust enough to not have replay attacks from the current game, given the implementation of Session keys, we are still vulnerable to stale messages from a previous round being sent as a new message in the following rounds. The system is thus, partially susceptible to Man-in-the-middle attack.

Our system does not have means to detect changes in data that might occur as a result of non-human caused events such as noise in the air or server crashes that can cause unintentional Denial of Service attacks. This can be a massive breach to data integrity as our server is not robust enough, therefore our server is susceptible to Denial of Service attacks.

5. Conclusion and Future Work

5.1 Conclusion

A simple version of multiplayer online poker game has been implemented. The server accepts two to five players and replicates a poker table. The poker version of five-card draw was executed for simplicity. The focus of the project was to create a system strong enough to counter most common security threats to which online games are prone. To keep the user data confidential, the popular cryptosystem techniques - RSA and AES were used. Session keys were established between the house and the players to encrypt communication and Public/Private Keys were used to sign and validate the bets. At the end of each game, the house announces the results and updates the wallets of all the players. To counter replay attacks, the session keys are then destroyed. We made use of Python cryptographic libraries to perform encryption and decryption.

5.2 Future scope

Despite the extent to which the project was implemented, there remains a lot of functionalities and features that can be added to improve the experience. If given more time and expertise, we would have liked to further develop and improve a few aspects of the project. The project does not include all of the features of poker. In certain edge cases where multiple players have the same hands and the pot needs to be split have not been considered. Measures to counter unwanted resource occupation by a client currently do not exist. Further security for the user-data confidentiality can be implemented, such as hashing the credentials before updating it in the database. Finally, in order to prevent server impersonation, X.905 server certificates can be used.

6. References

[1] V. H. H. Chen and Y. Wu, "Group identification as a mediator of the effect of players' anonymity on cheating in online games," Behavior & Information Technology, vol. 34, no. 7, pp. 658–667, Dec. 2013.

[2] C. Reilly, N. Smith, "Internet Gambling: an emerging field of Research," National Center for Responsible Gaming, Washington, D.C., USA, 2013. [Online]. Available: http://www.ncrg.org/sites/default/files/uploads/docs/white_papers/ncrg_wp_internetgambling_final.pdf

[3] L. Auriemma, D. Ferrante, “An overview of online poker security,” unpublished.
[4] J. Yan and B. Randell, “Security design in online games: from pong to online poker,” School of Computing Science, University of Newcastle upon Tyne, Newcastle upon Tyne, UK. CS-TR-889, Feb. 2005.