CPSC 452 - Crytography Final Project
Secure Poker

Members - Jake Cliff
- Scott Ng
- Kenny Chao
- Steven Tran

Abstract: In this project, we designed and implemented a secure poker application with node.js running as the server. We used node-forge for public key encryption to provide digital signatures and AES for confidentiality. For DSA, we used bigInt and crypto to allow large prime numbers.

To play the game, we need run our javascript with node.js as ‘node poker.js’ and have two instances of web browsers. Each browser must connect to a room. Each player must take turns to submit their choice for their hand. After 3 rounds, the game decides which player had the most wins.

Introduction: Upon entering the room, each player automatically generates a session key that will be encrypted by the server's public key. This ensures that no one other than the server and play will have access to the session key. Then, the server decrypts the encrypted session key with the private key. This key will be used with AES to encrypt messages passed between the server and client.

Design: When designing the look of the website did a simple google search for a template and found the one we wanted on W3School. Next was figuring out how to we wanted to do the handling of passing the variables between the pages, we accomplished this using POST. To get the appropriate variables to the HTML we passed them through javascript to populate the correct part of the pages. For the different stages of the game, we created multiple pages to handle this and saved time by using the same view with different pages.

Security Protocols: To accomplish digital signatures, the client has a choice of RSA or DSA. This was used to encrypt and decrypt the session key passed to the server. To accomplish confidentiality, we used AES which used the passed session key to encrypt the poker hand of the player.

Implementation: We developed our project in Javascript using Node.js as our front-end and back-end development. We utilized node-forge public-key encryption and AES for confidentiality.
Express and ejs were used to route HTML pages. Body-parser was used to parse the post request. For DSA, we used bigInt and crypto to implement the logic. When the server starts, and the player uses DSA, it randomly create the domain parameters (p,q,g). Then the client will randomly select a number between 1 and q-1 as his private key and will use that to calculate his public key and give that to the server. Then when the game starts and the player output his

hand to the server, it will sign the message using his private key and when the server receives the message, it will use his public key to verify the message has not been altered.

Conclusion: Overall, secure poker was accomplished with node-forge for digital signatures (RSA) and confidentiality (AES) and bigInt with crypto to complete DSA. The project runs on localhost:3000 with Node.js.