# Networking Final Project Design Document

Rachel Klesius and Garrett Moncrief

### Authentication Protocol

The authentication protocol used a protocol similar to Authentication Protocol ap4.0 from Kurose and Ross. In their version, a client sends to the server their identity, then the server sends a random number (or nonce) and then the client encrypts the random number using their symmetric key (Kurose 625). If the nonce the server generated is equivalent to the message the client sent back after decryption, then the client is who they say they are. The modification made was to use a Caesar cypher to encrypt a string of four random characters. This allowed for even greater security than one random number nonce, and eliminated the possibility of a playback attack. The protocol designed could use any encryption method, but the Caesar cypher used includes a shared secret between the client and server (in other words, a symmetric key) that corresponded to the number added to the letters to encrypt the message.

### Support for Multiple Clients

Through the use of the Caesar cypher it was not necessary to ensure that each client had a unique shared secret, and it was verified that several clients could utilize the authentication process simultaneously.

### DoS Protection

The server protocol relies on an ip logging system for protection from denial of service protection. Variable sized arrays store the client’s IP address and clock time when a new connection is accepted. If the client’s address has already been logged the current clock time is compared to the stored access time. If it has been less than 3000 milliseconds then the server skips any additional authentication and does not establish a weblite server, meaning that wasted resource allocation does not occur.

The rationale behind this is that denial of service attacks attempt to generate enough traffic to cripple the service. If the system has an access rate of R bps this means that minimal resources are being utilized by the offending attackers, thus increasing the R value and also the number and frequency of attackers in order to interrupt the system.[[1]](#footnote-1)

### Server

For our server implementation we chose to use the weblite1 server supplied by the instructor. The reason for this was that weblite1 already only supports only GET and was designed with the threaded access to the GET handling functions needed.

In anticipation of allowing multiple users to access the server there is a switch that opens it on different ports. This decision was made due to binding issues when multiple or subsequent users attempted to access the server using the same port number.

### Multithreading

Ideally our server client would utilize the Windows API process threads to open and terminate threads calling on the server for users. Unfortunately there was difficulty implementing this function, however our code is noted where it would’ve been implemented in an ideal case.

1. Kurose, J. F., & Ross, K. W. (2017). *Computer networking: A top-down approach*. Boston: Pearson. Pg 57. [↑](#footnote-ref-1)