**Team 17 Methodology, Anonychat**

Ben Korza, Mark Fitzgibbon, Mitchell Caisse

The Anonychat program will consist of a client/server program to be used by individual users, as well as a centralized name server to confirm peers to connect are online. The project will be written in C, for Linux Operating Systems, other operating systems is not a focus for our project. We will implement our chat program using a distributed client, in which clients will broadcast their messages to their peers. The messages will also be encrypted, this will allow for client messages to remain anonymous and their contents secure.

In order to do this we will need a centralized name server that will keep a list of the clients currently connected to the chat program and have the ability to distribute a list of peers to clients. This will allow new clients learn their peers and be able to start communicating with others. This will be accomplished by, when a client initial starts, it will initiate a TCP connection with the name server. While this connection is active, the name server will know that this client is connected, and be able to tell other clients that he is available as a peer. The name server will respond to this initial connection, with a list of a certain number of peers, around 5-10, if available. Each of these peers will also have a TTL associated with them. Once the TTL expires, the client will request a new peer, or peers from the name server. When the client terminates its connection with the name server, the name server will remove it from its list of active clients, and no longer distribute the client as a peer.

A client needs two things to communicate with others, first is the list of peers discussed above, second is public and private keys. We have decided that providing a platform to exchange and generate public/private keys is out of the scope of our project, and users will be responsible for generating their own public/private keys and distributing the public key to whom they wish to communicate with.

Once clients have a list of peers, and the public key of the user they wish to talk to, they can follow the proceeding steps to send a message to them.

If Alice wants to send a message to Bob, Alice’s client will:

1. Craft a message.
2. Encrypt it using Bob's public key.
3. Send the message to all of its known peers.

The N peers receiving a message will need to determine if the encoded message they received was intended for them to read or to be sent out and if they have seen the message before. When a message is received, a client will make an MD5 hash of the received message, and check against an internal list of hashes. If the hash does exist, then the message is not new to the client and nothing will be done, since it has already been processed. This will reduced network usage by not having a client forward a message if it has already seen it, making the chance of a cycle zero. If the hash proves to be new, it is added to the list, along with a time to live, and the client will decide if the message was intended for itself or to forward it on. To determine if the message was meant for it, the client will attempt to decrypt it with all of its private keys.

A successful decryption, determined by a sentinel string in the message, will imply that the client is meant to display the message to the user. If the message was unable to be decrypted, it will be passed along to the clients set of peers. The specific number of peers in broadcast range will need to be determined when a message is being sent based on what the client knows exists.

The message protocol that will be used for communication between clients as well as clients and the name server can be seen below. This protocol is based upon the IRC protocol, and will have a maximum length of 512 bytes. It is terminated by CRLF, which is included in the message size. It will also be prefixed with a sentinel (encryption prefix) when being sent between clients, the name server will not have one.

ENCRYPTION\_PREFIX [<prefix>]<command> <params> <...> <crlf>