client.hpp

```
#pragma once

#include <SFML/Network.hpp>
#include <boost/multiprecision/cpp_int.hpp>
#include <thread>
#include <mutex>

typedef boost::multiprecision::cpp_int bigint;

using boost::multiprecision::powm;

//bool tryLogIn(const string& username, const string& password, string&
error);

bool sendUDP(sf::UdpSocket& socket, const string& message, string&
error);
bool recvUDP(sf::UdpSocket& socket, void*& buffer, int& buffer_size);

void printBytes(const unsigned char* pBytes, const uint32_t nBytes);
void printBytes(void* pBytes, const uint32_t nBytes);


//Encryption
string encryptAES(const std::string& plaintext, unsigned char* key,
string& error);
string decryptAES(const string& ciphertext, unsigned char* key, string&
error);

//string bigintToHexString(const bigint& number);
void bigintToBytes(bigint key, unsigned char* buffer);

class Client
{
private:
    sf::TcpSocket tcp_socket;
    sf::UdpSocket udp_socket;
    sf::IpAddress udp_sender_address, udp_listener_address;
    string ip;
    unsigned short udp_sender_port, udp_listener_port;

    bigint p, g, secret;
    unsigned char key_bytes[16];


public:
    int player_id;
    string username;

    struct PlayerInfo
    {
        enum Flag
        {
            moving = 1,
            forward = 2,
            gun_shot = 4,
            quit = 8,
            got_shot = 16,
            dead = 32
        };
        int player_id;
        float dist2wall;
        float pos_x, pos_y, rot_x, rot_y;
        int flags;
        int score;
        char username[16];
    };


    Client();


    bool connectToServer(string& error);

    bool tryLogIn(const string& username, const string& password,
    string& error);
    bool trySignUp(const string& username, const string& password,
    string& error);

    bool sendEncryptedTCP(const string& msg, string& error);
    bool recvEncryptedTCP(void*& buffer, int& bufferSize, string&
    error);
    static bool recvTCP(sf::TcpSocket& socket, void*& buffer, int&
    buffer_size);
    static bool sendTCP(sf::TcpSocket& socket, const string& message,
    string& error);

    bool sendEncryptedUDP(void* buffer, int size, string& error);
    bool recvEncryptedUDP(void*& buffer, int& bufferSize, string&
    error);
    bool sendUDP(const string& message, string& error);
    bool recvUDP(void*& buffer, int& buffer_size);
    bool connected = false;
};
```