experiment/eccKEM.cpp

```cpp
/**
 * ECC KEM Benchmark
 * Author: Arnab Ghosh
 * Date: 11/6/2023
 *
 * Utilizes an Elliptic-Curve Cryptographic Scheme as a KEM (Key Encapsulation Mechanism)
 * Shall use as baseline from which other algorithms are compared
 *
 * In particular, this file takes advantage of ECIES (Elliptic Curve Integrated Encryptin Scheme)
 * Notably, ECIES combines both a KEM and a DEM (Data Encapsulation Mechanism).
 * ECIES is chosen as it is robustly standardized and thus trusted in rigorous security requirements.
 *
 * Note that this implementation took heavy reference from CryptoPP's Wiki page here: https://cryptopp.com/wiki/Elliptic_Curve_Integrated_Encryption_Scheme
*/

#include <iostream>
using std::ostream;
using std::cout;
using std::endl;

#include <string>
using std::string;

#include <cryptopp/files.h>
using CryptoPP::FileSink;
using CryptoPP::FileSource;

#include <cryptopp/hex.h>
using CryptoPP::HexEncoder;

#include <cryptopp/filters.h>
using CryptoPP::StringSink;
using CryptoPP::StringSource;
using CryptoPP::PK_EncryptorFilter;
using CryptoPP::PK_DecryptorFilter;

#include <cryptopp/osrng.h>
using CryptoPP::AutoSeededRandomPool;

#include <cryptopp/integer.h>
using CryptoPP::Integer;

#include <cryptopp/pubkey.h>
using CryptoPP::PublicKey;
using CryptoPP::PrivateKey;

#include <cryptopp/eccrypto.h>
using CryptoPP::ECP;    // Prime field
using CryptoPP::EC2N;   // Binary field
using CryptoPP::ECIES;
using CryptoPP::ECPPoint;
using CryptoPP::DL_GroupParameters_EC;
```

```
using CryptoPP::DL_GroupPrecomputation;
using CryptoPP::DL_FixedBasePrecomputation;

#include <cryptopp/pubkey.h>
using CryptoPP::DL_PrivateKey_EC;
using CryptoPP::DL_PublicKey_EC;

#include <cryptopp/asn.h>
#include <cryptopp/oids.h>
namespace ASN1 = CryptoPP::ASN1;

#include <cryptopp/cryptlib.h>
using CryptoPP::PK_Encryptor;
using CryptoPP::PK_Decryptor;
using CryptoPP::g_nullNameValuePairs;

/**
 * Saves the private given key to a file of a given name.
*/
void savePrivateKey(const PrivateKey& key, const string& file) {
    FileSink sink(file.c_str());
    key.Save(sink);
}

/**
 * Saves the public given key to a file of a given name.
 * Note that this is different than void SavePrivateKey because PrivateKey and PublicKey are different data types in CryptoPP;
 * Regardless, the implementation for both methods are exactly the same. An Interface would be quite nice in this scenario, would it not?
*/
void savePublicKey(const PublicKey& key, const string& file) {
    FileSink sink(file.c_str());
    key.Save(sink);
}

/**
 * Loads a private key from a given file into the given reference.
*/
void loadPrivateKey(PrivateKey& key, const string& file) {
    FileSource source(file.c_str(), true);
    key.Load(source);
}

/**
 * Loads a public key from a given file into the given reference.
*/
void loadPublicKey(PublicKey& key, const string& file) {
    FileSource source(file.c_str(), true);
    key.Load(source);
}


/**
 * Entry point of program
*/
int main(int argc, char* argv[]) {
```

```
    // Parse arguments
    char* payload = argv[1];

    // Random number generator
    AutoSeededRandomPool randomNumberGenerator;

    // Encryptor and Decryptor Instances for Key Generation
    ECIES<ECP>::Decryptor privateKeyGenerator (randomNumberGenerator, ASN1::secp256r1());
    ECIES<ECP>::Encryptor publicKeyGenerator (privateKeyGenerator);

    // We now fetch and store private keys
    savePrivateKey(privateKeyGenerator.GetPrivateKey(), "ECCKEM_PRIVATEKEY");
    savePublicKey(publicKeyGenerator.GetPublicKey(), "ECCKEM_PUBLICKEY");

    // Load the keys
    // First we create our Encryptor and Decryptor instances
    ECIES<ECP>::Decryptor decryptor;
    ECIES<ECP>::Encryptor encryptor;

    loadPrivateKey(decryptor.AccessPrivateKey(), "ECCKEM_PRIVATEKEY");
    loadPublicKey(encryptor.AccessPublicKey(), "ECCKEM_PUBLICKEY");

    // We now begin the encryption and decryption process
    string encryptedMessageString;
    FileSource encryptedMessage (payload, true, new PK_EncryptorFilter(randomNumberGenerator,
publicKeyGenerator, new StringSink(encryptedMessageString)));
    StringSource encryptedMessageFile (encryptedMessageString, true, new FileSink("ECCKEM_OUTPUT"));
}

```