OpenFHE Code Example

OpenFHE Code Example

```
CCParams<CryptoContextCKKSRNS> parameters;
parameters.SetMultiplicativeDepth(10);
parameters.SetScalingModSize(50);
parameters.SetBatchSize(2);

CryptoContext<DCRTPoly> cc = GenCryptoContext(parameters);

cc->Enable(PKE);
cc->Enable(KEYSWITCH);
cc->Enable(LEVELEDSHE);

auto keys = cc->KeyGen();
cc->EvalMultKeyGen(keys.secretKey);
```

Initialization of the security parameters:
`MultiplicativeDepth:` The number of times a multiplication is possible on the ciphertexts.
`ScalingModSize:` The bit-length of the scaling factor (determines accuracy of the result)
`BatchSize:` The length of the input vector.

Generation of the CryptoContext. It's the interface to all functions of the OpenFHE library and stores all parameters and the eval key.

Features we want to be enabled.

Key generation. The eval key is saved in the CryptoContext

OpenFHE Code Example

```
std::vector<double> x1 = {0.25, 0.5};
std::vector<double> x2 = {5.0, 4.0};

Plaintext ptxt1 = cc->MakeCKKSPackedPlaintext(x1);
Plaintext ptxt2 = cc->MakeCKKSPackedPlaintext(x2);

auto c1 = cc->Encrypt(keys.publicKey, ptxt1);
auto c2 = cc->Encrypt(keys.publicKey, ptxt2);

auto cAdd = cc->EvalAdd(c1, c2);
auto cMul = cc->EvalMult(c1, c2);

Plaintext result;
cc->Decrypt(keys.secretKey, cAdd, &result);
result->SetLength(batchSize);
std::cout << "x1 + x2 = " << result << std::endl;

cc->Decrypt(keys.secretKey, cMul, &result);
result->SetLength(batchSize);
std::cout << "x1 * x2 = " << result << std::endl;
```

Initialization of the input vectors

Encoding

Encryption

Applying Multiplication and Addition

Decrypting and printing the results