CKKS Scheme Based Fully Homomorphic Encryption

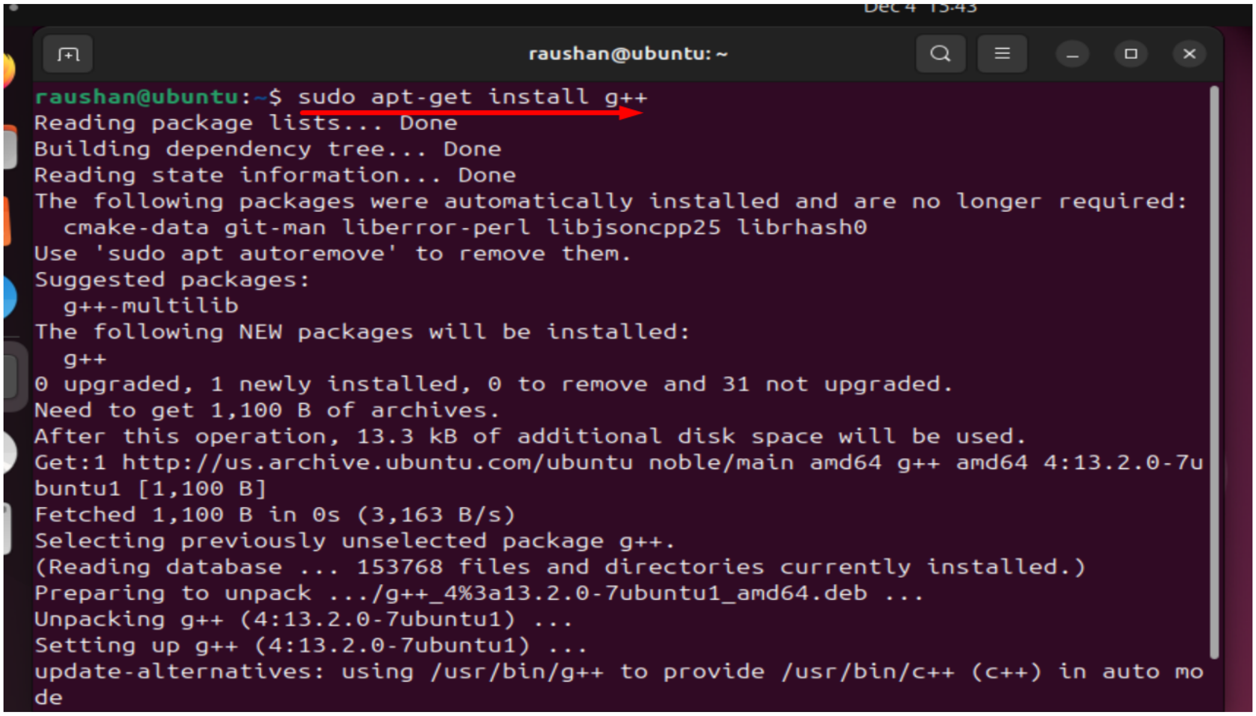
1. **Environment Set Up for CKKS.**
   CKKS is a fully homomorphic scheme. It is based on approximate arithmetic-based computation on encrypted data. Operations such as addition, subtraction, multiplication, and rotation can be done with this scheme. We will see some basic arithmetic operations on encrypted data. Before that, we must set up the lab running environment. There are very few specific libraries to implement CKKS. Here, we will set up Open FHE development environment. It has been implemented in standard C++ programming language. It will soon be available in python too. One can set up the environment in any operating system. However, Unix based OS would be apt for this scheme. You can use any Unix system. Here the demo has been done in Ubuntu Linux 24.04 (64-bit).

A. **Install Ubuntu Linux (You can use virtual box)**

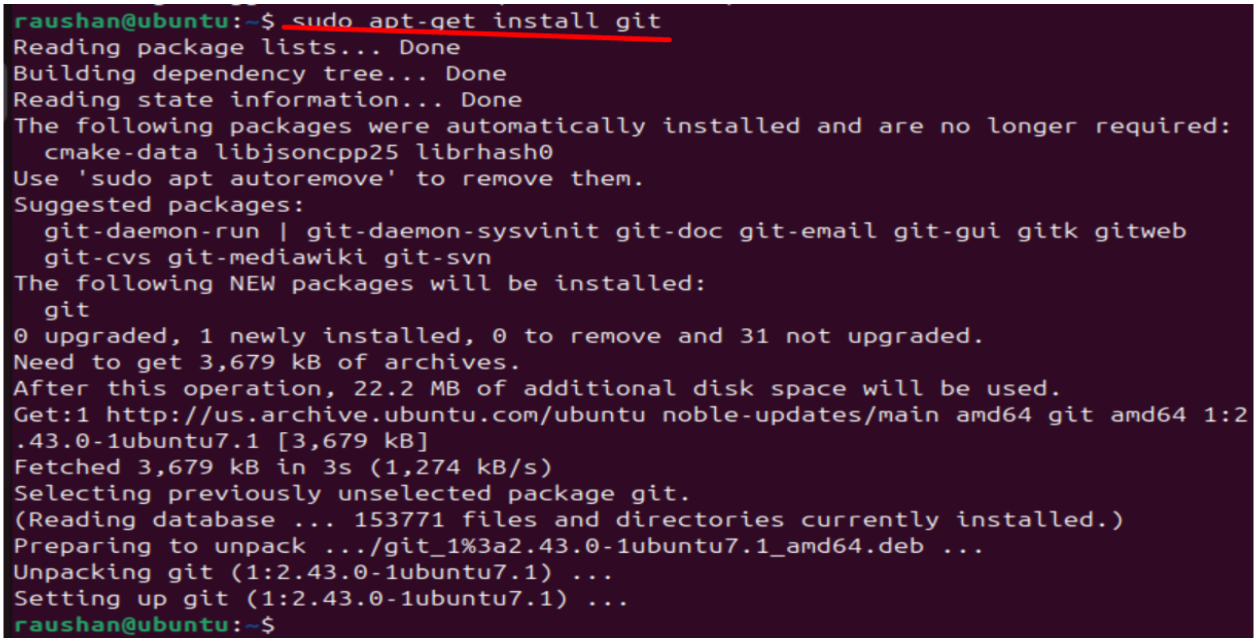
B. **You can follow the link below link and install it yourself. Though, I am showing you step by step process to set up.**

(https://openfhe-development.readthedocs.io/en/latest/sphinx_rsts/intro/installation/linux.html)

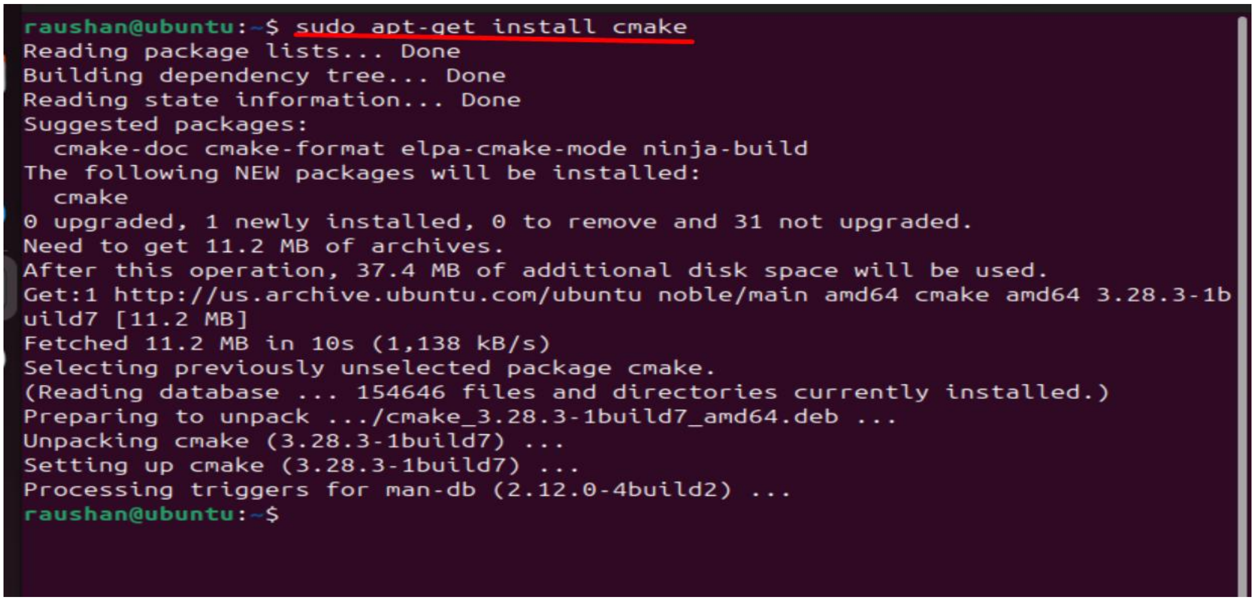
C. **Install g++ (C++ compiler), git, & cmake (.exe builder)**



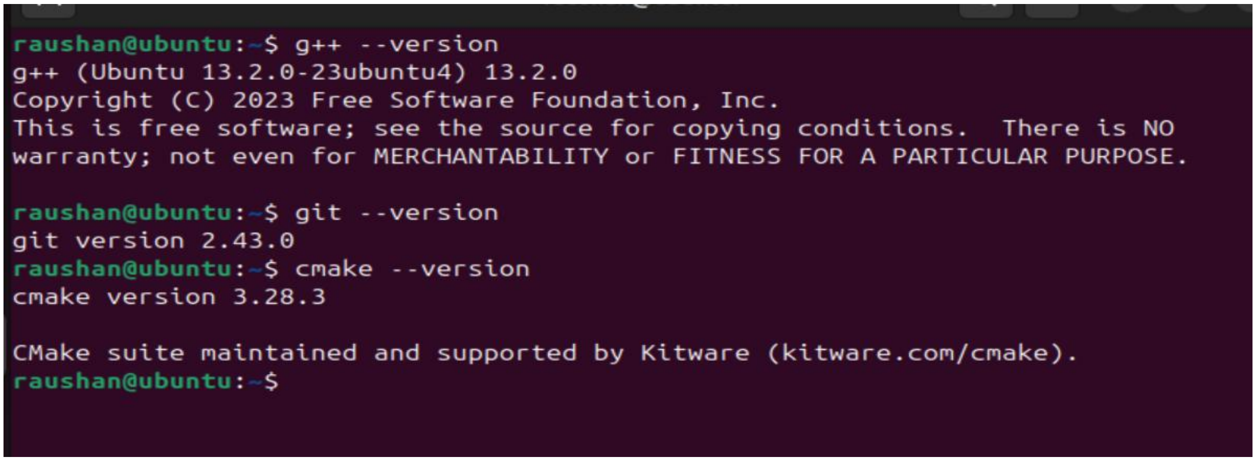
```
raushan@ubuntu:~$ sudo apt-get install g++
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  cmake-data git-man liberror-perl libjsoncpp25 librhash0
Use 'sudo apt autoremove' to remove them.
Suggested packages:
  g++-multilib
The following NEW packages will be installed:
  g++
0 upgraded, 1 newly installed, 0 to remove and 31 not upgraded.
Need to get 1,100 B of archives.
After this operation, 13.3 kB of additional disk space will be used.
Get:1 http://us.archive.ubuntu.com/ubuntu noble/main amd64 g++ amd64 4:13.2.0-7u
buntu1 [1,100 B]
Fetched 1,100 B in 0s (3,163 B/s)
Selecting previously unselected package g++.
(Reading database ... 153768 files and directories currently installed.)
Preparing to unpack .../g++_4%3a13.2.0-7ubuntu1_amd64.deb ...
Unpacking g++ (4:13.2.0-7ubuntu1) ...
Setting up g++ (4:13.2.0-7ubuntu1) ...
update-alternatives: using /usr/bin/g++ to provide /usr/bin/c++ (c++) in auto mo
de
```



```
raushan@ubuntu:~$ sudo apt-get install git
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  cmake-data libjsoncpp25 librhash0
Use 'sudo apt autoremove' to remove them.
Suggested packages:
  git-daemon-run | git-daemon-sysvinit git-doc git-email git-gui gitk gitweb
  git-cvs git-mediawiki git-svn
The following NEW packages will be installed:
  git
0 upgraded, 1 newly installed, 0 to remove and 31 not upgraded.
Need to get 3,679 kB of archives.
After this operation, 22.2 MB of additional disk space will be used.
Get:1 http://us.archive.ubuntu.com/ubuntu noble-updates/main amd64 git amd64 1:2
.43.0-1ubuntu7.1 [3,679 kB]
Fetched 3,679 kB in 3s (1,274 kB/s)
Selecting previously unselected package git.
(Reading database ... 153771 files and directories currently installed.)
Preparing to unpack .../git_1%3a2.43.0-1ubuntu7.1_amd64.deb ...
Unpacking git (1:2.43.0-1ubuntu7.1) ...
Setting up git (1:2.43.0-1ubuntu7.1) ...
raushan@ubuntu:~$
```

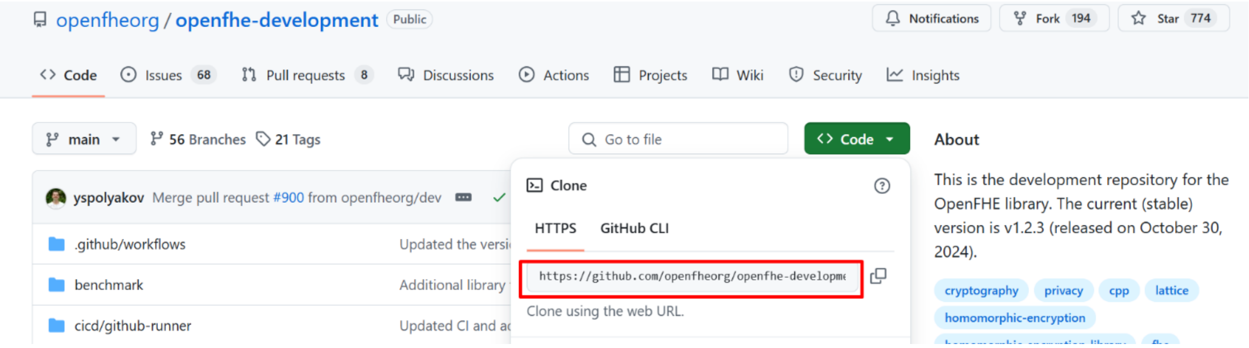


```
raushan@ubuntu:~$ sudo apt-get install cmake
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Suggested packages:
  cmake-doc cmake-format elpa-cmake-mode ninja-build
The following NEW packages will be installed:
  cmake
0 upgraded, 1 newly installed, 0 to remove and 31 not upgraded.
Need to get 11.2 MB of archives.
After this operation, 37.4 MB of additional disk space will be used.
Get:1 http://us.archive.ubuntu.com/ubuntu noble/main amd64 cmake amd64 3.28.3-1b
uild7 [11.2 MB]
Fetched 11.2 MB in 10s (1,138 kB/s)
Selecting previously unselected package cmake.
(Reading database ... 154646 files and directories currently installed.)
Preparing to unpack .../cmake_3.28.3-1build7_amd64.deb ...
Unpacking cmake (3.28.3-1build7) ...
Setting up cmake (3.28.3-1build7) ...
Processing triggers for man-db (2.12.0-4build2) ...
raushan@ubuntu:~$
```

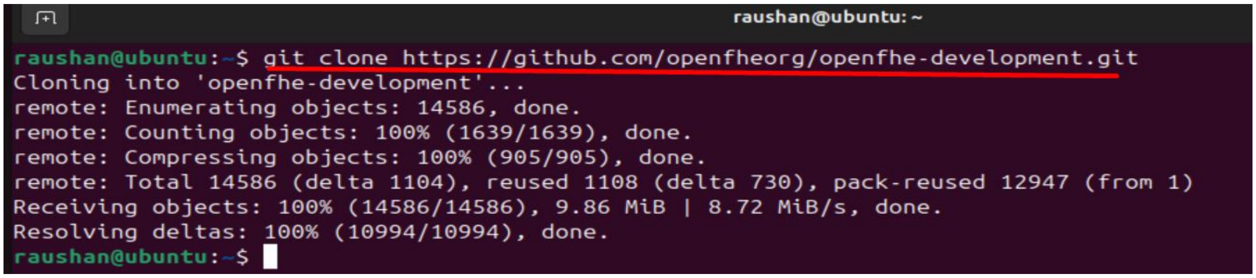


```
raushan@ubuntu:~$ g++ --version
g++ (Ubuntu 13.2.0-23ubuntu4) 13.2.0
Copyright (C) 2023 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

raushan@ubuntu:~$ git --version
git version 2.43.0
raushan@ubuntu:~$ cmake --version
cmake version 3.28.3

CMake suite maintained and supported by Kitware (kitware.com/cmake).
raushan@ubuntu:~$
```

D. Now we need to clone the openfhe-development repository.

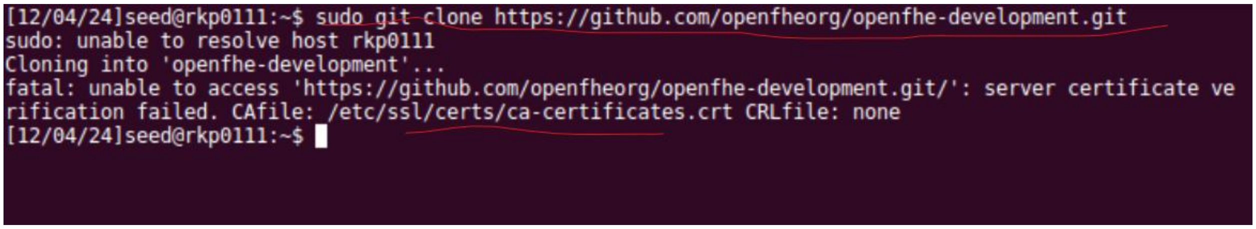


This is the development repository for the OpenFHE library. The current (stable) version is v1.2.3 (released on October 30, 2024).

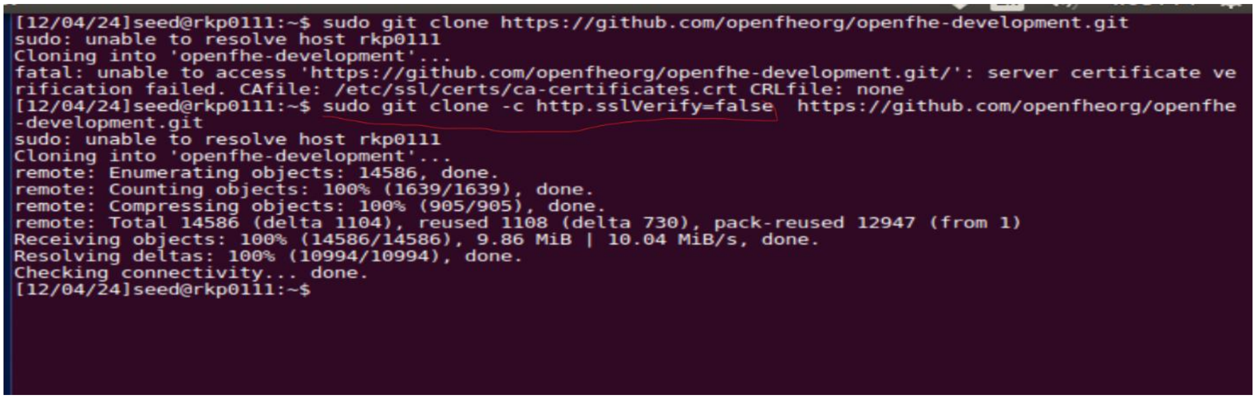


```
raushan@ubuntu:~$ git clone https://github.com/openfheorg/openfhe-development.git
Cloning into 'openfhe-development'...
remote: Enumerating objects: 14586, done.
remote: Counting objects: 100% (1639/1639), done.
remote: Compressing objects: 100% (905/905), done.
remote: Total 14586 (delta 1104), reused 1108 (delta 730), pack-reused 12947 (from 1)
Receiving objects: 100% (14586/14586), 9.86 MiB | 8.72 MiB/s, done.
Resolving deltas: 100% (10994/10994), done.
raushan@ubuntu:~$
```

#Sometimes you might get SSL certificates error in ubuntu. Resolve by doing Google or you might try following way.



```
[12/04/24]seed@rkp0111:~$ sudo git clone https://github.com/openfheorg/openfhe-development.git
sudo: unable to resolve host rkp0111
Cloning into 'openfhe-development'...
fatal: unable to access 'https://github.com/openfheorg/openfhe-development.git/': server certificate ve
rification failed. CAfile: /etc/ssl/certs/ca-certificates.crt CRLfile: none
[12/04/24]seed@rkp0111:~$
```

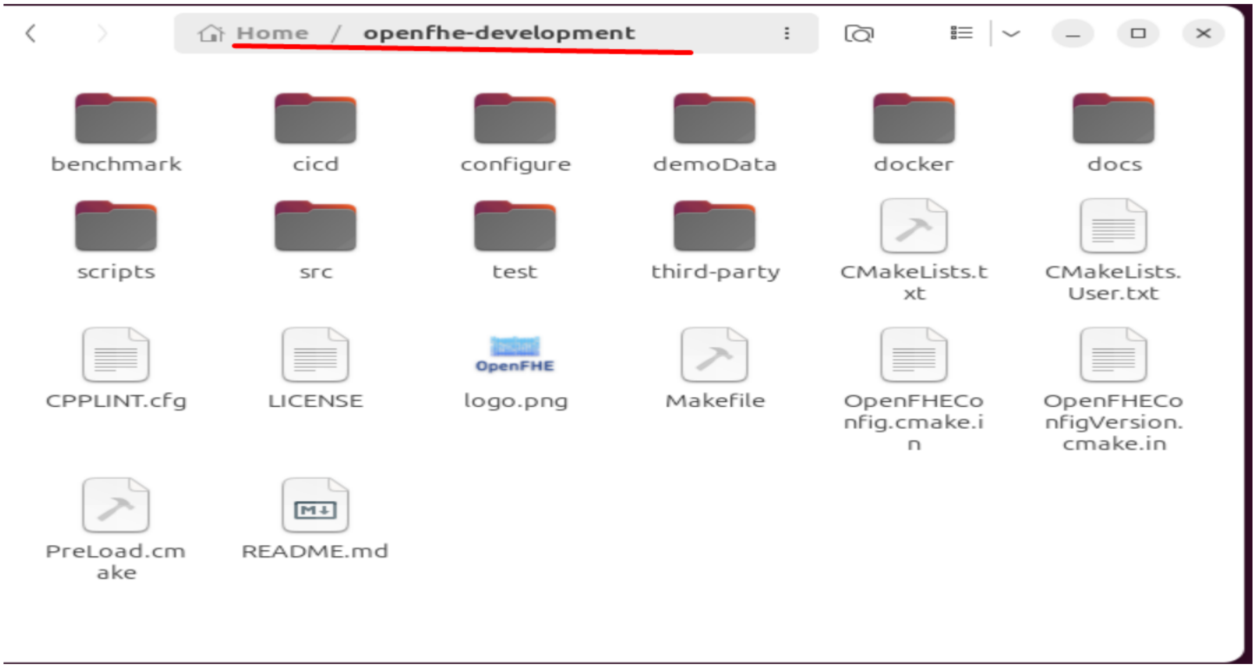


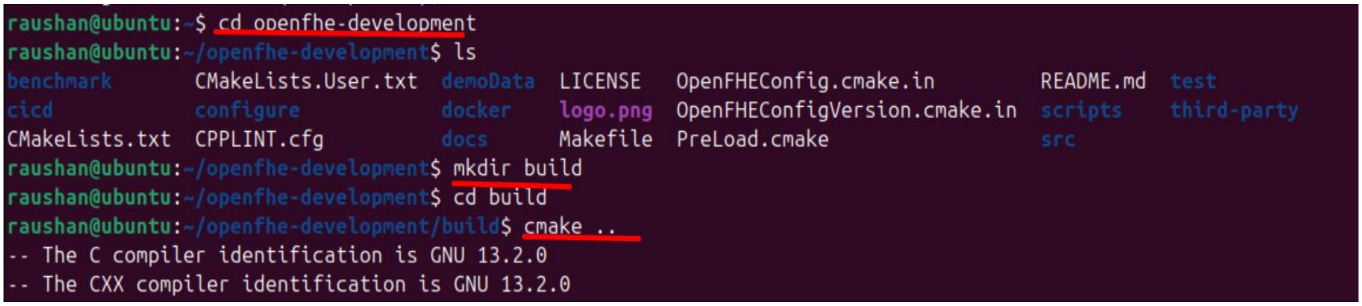
```
[12/04/24]seed@rkp0111:~$ sudo git clone https://github.com/openfheorg/openfhe-development.git
sudo: unable to resolve host rkp0111
Cloning into 'openfhe-development'...
fatal: unable to access 'https://github.com/openfheorg/openfhe-development.git/': server certificate ve
rification failed. CAfile: /etc/ssl/certs/ca-certificates.crt CRLfile: none
[12/04/24]seed@rkp0111:~$ sudo git clone -c http.sslVerify=false  https://github.com/openfheorg/openfhe
-development.git
sudo: unable to resolve host rkp0111
Cloning into 'openfhe-development'...
remote: Enumerating objects: 14586, done.
remote: Counting objects: 100% (1639/1639), done.
remote: Compressing objects: 100% (905/905), done.
remote: Total 14586 (delta 1104), reused 1108 (delta 730), pack-reused 12947 (from 1)
Receiving objects: 100% (14586/14586), 9.86 MiB | 10.04 MiB/s, done.
Resolving deltas: 100% (10994/10994), done.
Checking connectivity... done.
[12/04/24]seed@rkp0111:~$
```

E. **From the Openfhe-development repository, run following commands one by one.**

We are trying to create another build directory inside openfhe-development, where binary will be built. If not work, try "sudo" too.

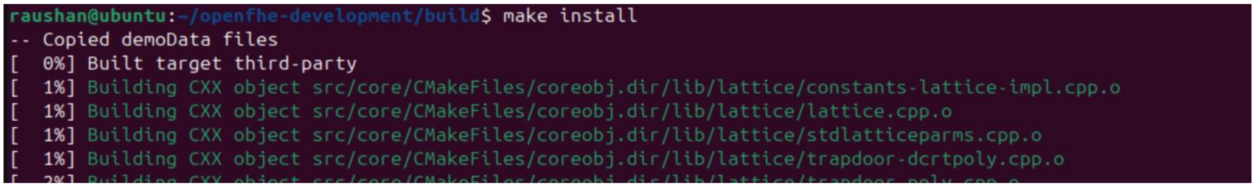
>>mkdir build
>>cd build
>>cmake ..
>>make (This will take 30-50 minutes)
>>make install (This is just for check up the production environment)





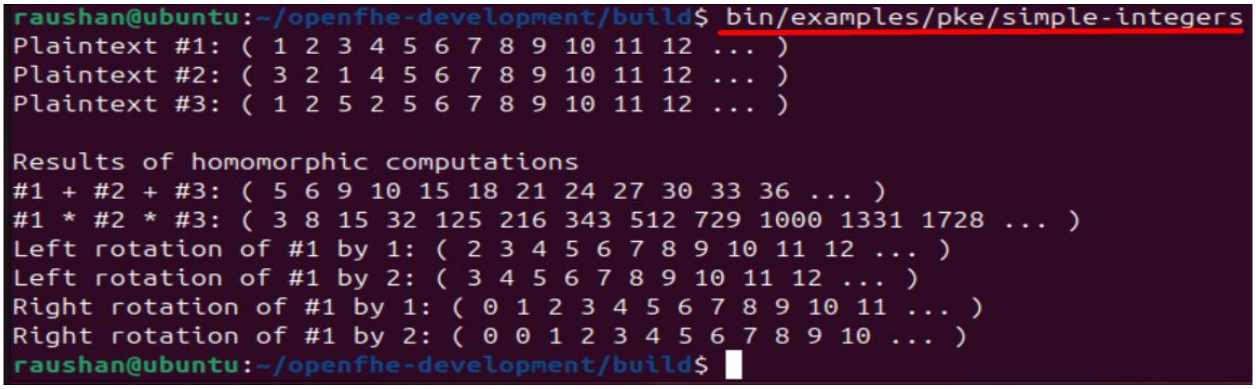
```
raushan@ubuntu:~$ cd openfhe-development
raushan@ubuntu:~/openfhe-development$ ls
benchmark       CMakeLists.User.txt  demoData  LICENSE   OpenFHEConfig.cmake.in         README.md  test
cicd            configure            docker    logo.png  OpenFHEConfigVersion.cmake.in  scripts    third-party
CMakeLists.txt  CPPLINT.cfg          docs      Makefile  PreLoad.cmake                  src
raushan@ubuntu:~/openfhe-development$ mkdir build
raushan@ubuntu:~/openfhe-development$ cd build
raushan@ubuntu:~/openfhe-development/build$ cmake ..
-- The C compiler identification is GNU 13.2.0
-- The CXX compiler identification is GNU 13.2.0
```

Note- Below Screenshot is wrong. First run "make" or "sudo make" then "make install" only.



```
raushan@ubuntu:~/openfhe-development/build$ make install
-- Copied demoData files
[  0%] Built target third-party
[  1%] Building CXX object src/core/CMakeFiles/coreobj.dir/lib/lattice/constants-lattice-impl.cpp.o
[  1%] Building CXX object src/core/CMakeFiles/coreobj.dir/lib/lattice/lattice.cpp.o
[  1%] Building CXX object src/core/CMakeFiles/coreobj.dir/lib/lattice/stdlatticeparms.cpp.o
[  1%] Building CXX object src/core/CMakeFiles/coreobj.dir/lib/lattice/trapdoor-dcrtpoly.cpp.o
[  2%] Building CXX object src/core/CMakeFiles/coreobj.dir/lib/lattice/trapdoor-poly.cpp.o
```

F. **Run the built-in example to check whether you have installed successfully or not.**



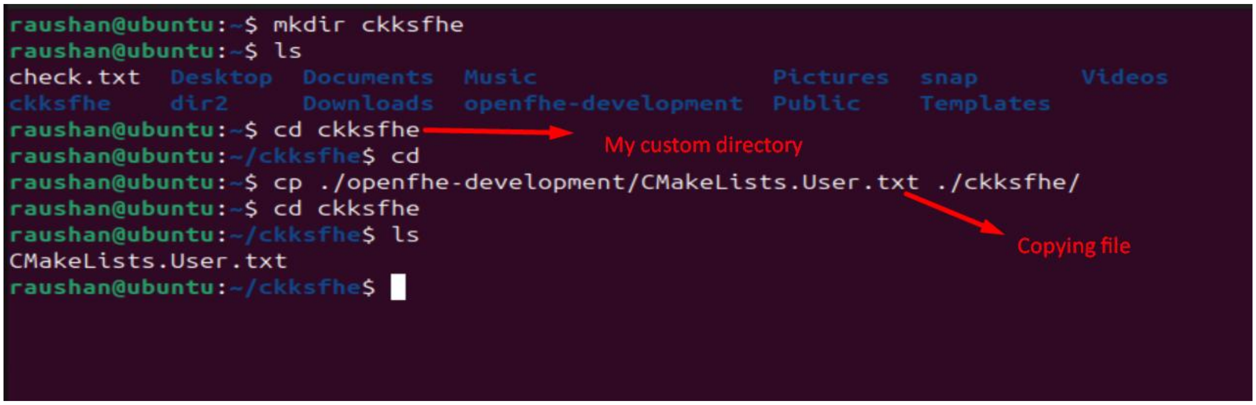
```
raushan@ubuntu:~/openfhe-development/build$ bin/examples/pke/simple-integers
Plaintext #1: ( 1 2 3 4 5 6 7 8 9 10 11 12 ... )
Plaintext #2: ( 3 2 1 4 5 6 7 8 9 10 11 12 ... )
Plaintext #3: ( 1 2 5 2 5 6 7 8 9 10 11 12 ... )

Results of homomorphic computations
#1 + #2 + #3: ( 5 6 9 10 15 18 21 24 27 30 33 36 ... )
#1 * #2 * #3: ( 3 8 15 32 125 216 343 512 729 1000 1331 1728 ... )
Left rotation of #1 by 1: ( 2 3 4 5 6 7 8 9 10 11 12 ... )
Left rotation of #1 by 2: ( 3 4 5 6 7 8 9 10 11 12 ... )
Right rotation of #1 by 1: ( 0 1 2 3 4 5 6 7 8 9 10 11 ... )
Right rotation of #1 by 2: ( 0 0 1 2 3 4 5 6 7 8 9 10 ... )
raushan@ubuntu:~/openfhe-development/build$
```

It means the installation has been done correctly. If you can't get something like above, you might make mistakes in the above steps.

G. **How to Build your Project and run the code successfully?**

a. Here, you need to create your own directory where you can make your code file, .exe files. Make sure this directory will not be inside openfhe-development directory. **Example: - "ckksfhe" directory where we put our everything.**

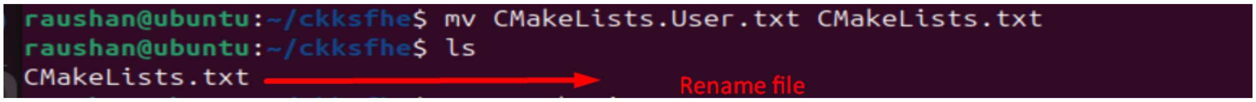
b. Copy "CMakeLists.User.txt" from openfhe-development directory to your newly created custom directory.



```
raushan@ubuntu:~$ mkdir ckksfhe
raushan@ubuntu:~$ ls
check.txt  Desktop  Documents  Music                Pictures  snap       Videos
ckksfhe    dir2     Downloads  openfhe-development  Public    Templates
raushan@ubuntu:~$ cd ckksfhe
raushan@ubuntu:~/ckksfhe$ cd
raushan@ubuntu:~$ cp ./openfhe-development/CMakeLists.User.txt ./ckksfhe/
raushan@ubuntu:~$ cd ckksfhe
raushan@ubuntu:~/ckksfhe$ ls
CMakeLists.User.txt
raushan@ubuntu:~/ckksfhe$
```

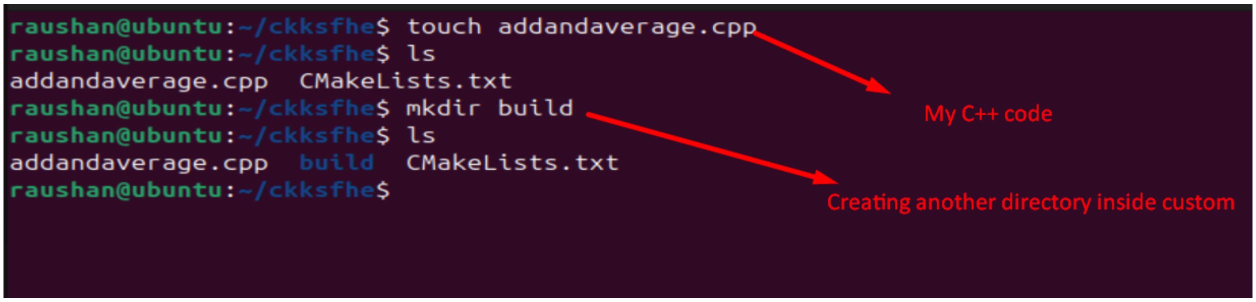
c. **Now rename "CMakeLists.User.txt" filename as "CMakeLists.txt" and to build your .exe from C++ code add such as**

"add_executable (name-of-binary   nameofc++file.cpp)"



```
raushan@ubuntu:~/ckksfhe$ mv CMakeLists.User.txt CMakeLists.txt
raushan@ubuntu:~/ckksfhe$ ls
CMakeLists.txt
```

d. **Create own c++ file inside directory ckksfhe, also create another directory "build" inside ckksfhe directory where binary of own c++ will be build.**



```
raushan@ubuntu:~/ckksfhe$ touch addandaverage.cpp
raushan@ubuntu:~/ckksfhe$ ls
addandaverage.cpp  CMakeLists.txt
raushan@ubuntu:~/ckksfhe$ mkdir build
raushan@ubuntu:~/ckksfhe$ ls
addandaverage.cpp  build  CMakeLists.txt
raushan@ubuntu:~/ckksfhe$
```

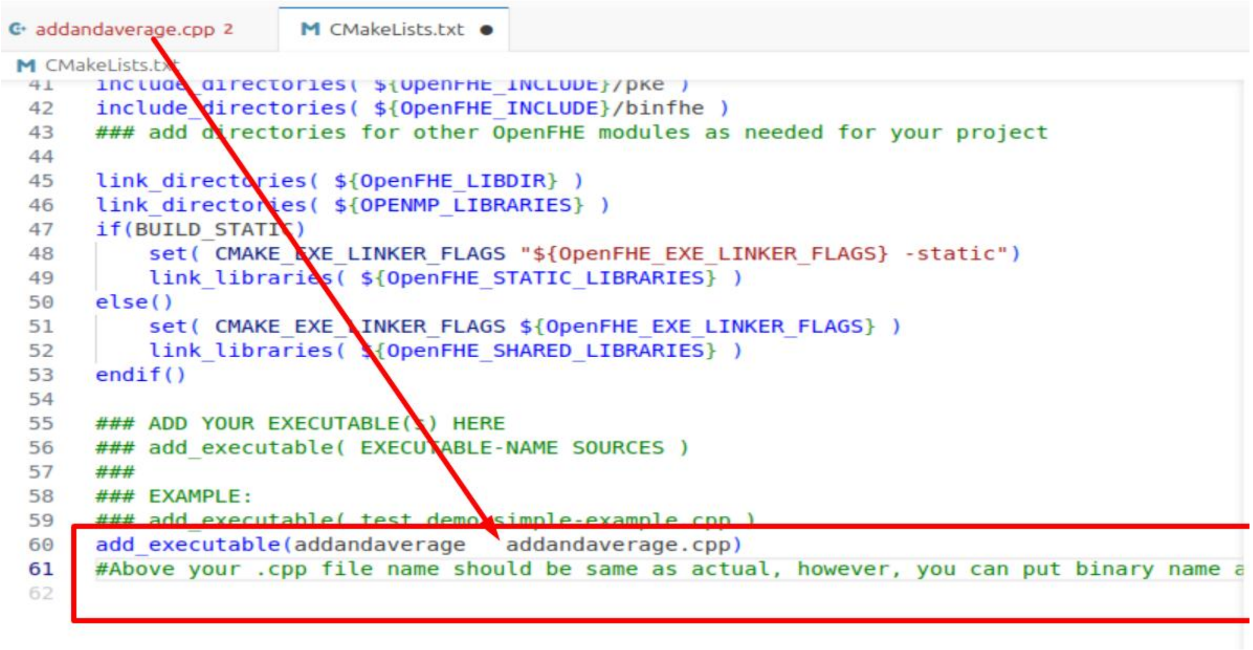
My C++ code

Creating another directory inside custom

2. **Lab to add & multiply two numbers and find average of two.**

   >>We need to write code to take two user defined number (float), find sum of two, multiply two of them, and find average of two.

- Write code and put it inside your custom directory. Example, in my case I have created ckksfhe directory, inside which I have created c++ file “addandaverage.cpp”.
- Edit CMakeLists.txt to create .exe binary for addandaverage.cpp code.

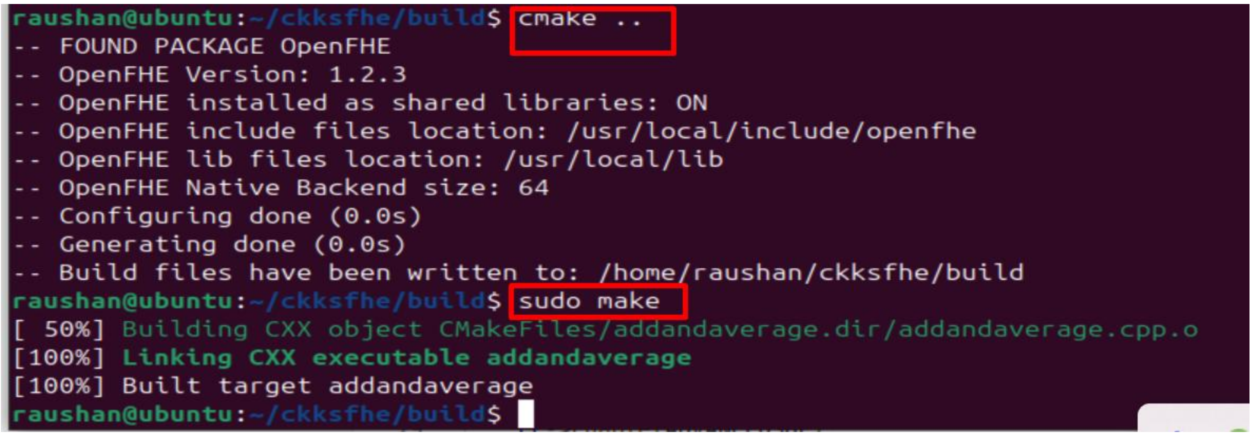


```
include_directories( ${OpenFHE_INCLUDE}/pke )
include_directories( ${OpenFHE_INCLUDE}/binfhe )
### add directories for other OpenFHE modules as needed for your project

link_directories( ${OpenFHE_LIBDIR} )
link_directories( ${OPENMP_LIBRARIES} )
if(BUILD_STATIC)
    set( CMAKE_EXE_LINKER_FLAGS "${OpenFHE_EXE_LINKER_FLAGS} -static")
    link_libraries( ${OpenFHE_STATIC_LIBRARIES} )
else()
    set( CMAKE_EXE_LINKER_FLAGS ${OpenFHE_EXE_LINKER_FLAGS} )
    link_libraries( ${OpenFHE_SHARED_LIBRARIES} )
endif()

### ADD YOUR EXECUTABLE(S) HERE
### add_executable( EXECUTABLE-NAME SOURCES )
###
### EXAMPLE:
### add_executable( test demo-simple-example.cpp )
add_executable(addandaverage   addandaverage.cpp)
#Above your .cpp file name should be same as actual, however, you can put binary name a
```

- From build directory of ckksfhe, run one by one following commands (You can with sudo too)

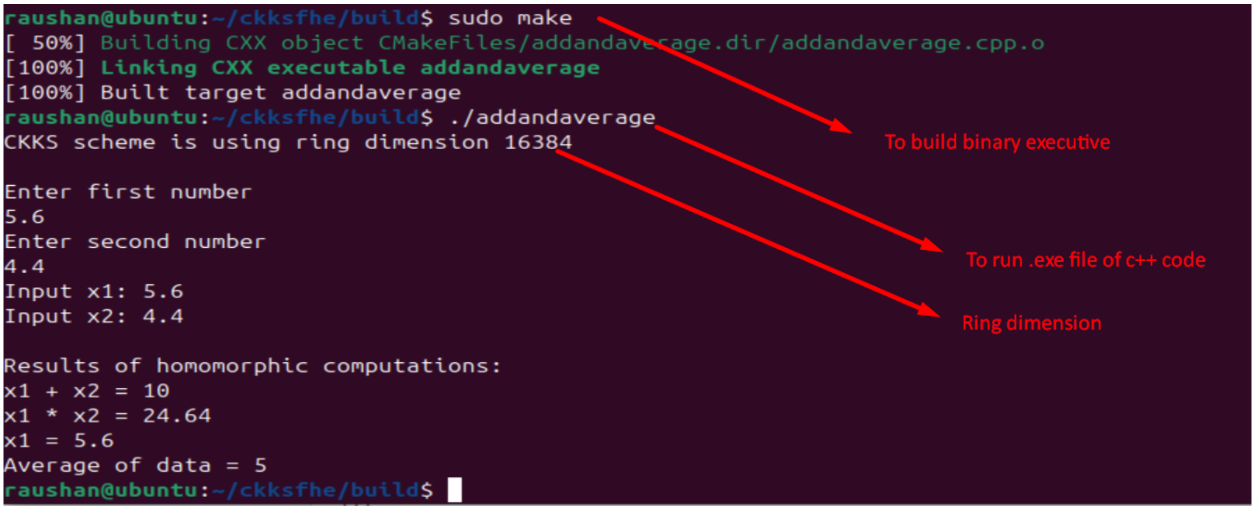
  >>cmake ..

  >>make



```
raushan@ubuntu:~/ckksfhe/build$ cmake ..
-- FOUND PACKAGE OpenFHE
-- OpenFHE Version: 1.2.3
-- OpenFHE installed as shared libraries: ON
-- OpenFHE include files location: /usr/local/include/openfhe
-- OpenFHE lib files location: /usr/local/lib
-- OpenFHE Native Backend size: 64
-- Configuring done (0.0s)
-- Generating done (0.0s)
-- Build files have been written to: /home/raushan/ckksfhe/build
raushan@ubuntu:~/ckksfhe/build$ sudo make
[ 50%] Building CXX object CMakeFiles/addandaverage.dir/addandaverage.cpp.o
[100%] Linking CXX executable addandaverage
[100%] Built target addandaverage
raushan@ubuntu:~/ckksfhe/build$
```

- **Now run .exe file to view output of our code**



```
raushan@ubuntu:~/ckksfhe/build$ sudo make
[ 50%] Building CXX object CMakeFiles/addandaverage.dir/addandaverage.cpp.o
[100%] Linking CXX executable addandaverage
[100%] Built target addandaverage
raushan@ubuntu:~/ckksfhe/build$ ./addandaverage
CKKS scheme is using ring dimension 16384

Enter first number
5.6
Enter second number
4.4
Input x1: 5.6
Input x2: 4.4

Results of homomorphic computations:
x1 + x2 = 10
x1 * x2 = 24.64
x1 = 5.6
Average of data = 5
raushan@ubuntu:~/ckksfhe/build$
```

Above, we have taken two inputs as float values, changed them into two vectors. Both vectors have been encrypted, then added & multiplied together. Even the average is calculated from encrypted data. Above, output might give you skeptical sense that only normal addition & multiplication of float data. So, Check the code part carefully.

Error: - "Error while loading shared libraries: libOPENFHEbinfhe.so.1: can't open shared file: No such file or directory

Solution: - Run "export LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:/usr/local/lib/" Or "export LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:/path of your installed library"

(https://openfhe.discourse.group/t/problem-installing-the-library-on-ubuntu-20-04/1077)

References: -

1) https://openfhe.discourse.group/ (Discourse page, post your issue to get help)
2) https://openfhe-development.readthedocs.io/en/latest/sphinx_rsts/intro/building_user_applications.html (OpenFhe documentation URL to install and build your .exe in different Operating system)
3) https://eprint.iacr.org/2016/421.pdf (Actual Paper of CKKS)
4) https://eprint.iacr.org/2020/1118 (OpenFhe library implemented paper)
5) https://docs.google.com/presentation/d/1YLhaLJrJZqwP8c_yVQCPQZlrDKY_b_Q9D_D5IVK0A5w/edit?usp=sharing (Slides of description about CKKS)
6) https://www.youtube.com/watch?v=iQlgeL64vfo&t=181s (Introduction video by author)