# **Securing API Keys in Android App using NDK**

**Step 1: Install the required tools**

* **NDK (Native Development Kit)**
* **LLDB (Low Level Debugger)**
* **CMake**

### Step 2: Create a (native-lib.cpp) file

Create a new folder,**cpp**, inside app/src/main.

Once you've created it, right-click on the cpp folder, click on **New → C/C++ Source File**, and name your file native-lib.

**Step 3: Store your API key inside the native-lib.cpp file**

Inside your native-lib.cpp, add the following code:

Multiple securing api keys

#include <jni.h>

#include <string>

extern "C" JNIEXPORT jstring

JNICALL

Java\_com\_package\_name\_Keys\_apiKey1(JNIEnv \*env, jobject object) {

std::string api\_key1= "your\_api\_key\_goes\_here";

Java\_com\_package\_name\_Keys\_apiKey2(JNIEnv \*env, jobject object) {

std::string api\_key2= "your\_api\_key\_goes\_here";

return env->NewStringUTF(api\_key2.c\_str());}

**Step 4: Create a CMakeLists.txt file**

Under the app/ folder, create a new text file and name it CMakeLists.txt. Add the following code in the file:

# For more information about using CMake with Android Studio, read the

# documentation: https://d.android.com/studio/projects/add-native-code.html

# Sets the minimum version of CMake required to build the native library.

cmake\_minimum\_required(VERSION 3.4.1)

# Creates and names a library, sets it as either STATIC

# or SHARED, and provides the relative paths to its source code.

# You can define multiple libraries, and CMake builds them for you.

# Gradle automatically packages shared libraries with your APK.

add\_library( # Sets the name of the library.

native-lib

# Sets the library as a shared library.

SHARED

# Provides a relative path to your source file(s).

src/main/cpp/native-lib.cpp )

# Searches for a specified prebuilt library and stores the path as a

# variable. Because CMake includes system libraries in the search path by

# default, you only need to specify the name of the public NDK library

# you want to add. CMake verifies that the library exists before

# completing its build.

find\_library( # Sets the name of the path variable.

log-lib

# Specifies the name of the NDK library that

# you want CMake to locate.

log )

# Specifies libraries CMake should link to your target library. You

# can link multiple libraries, such as libraries you define in this

# build script, prebuilt third-party libraries, or system libraries.

target\_link\_libraries( # Specifies the target library.

native-lib

# Links the target library to the log library

# included in the NDK.

${log-lib} )

### Step 5: Configure Gradle for CMake

Add the following lines to the android block in your app-level build.gradle file:

android {

externalNativeBuild {

cmake {

path "CMakeLists.txt"

}

}

}

**Step 6: Creating the Keys.kt file**

Create a Kotlin object, Keys.kt, and obtain your API key as followed:

object Keys {

init {

System.loadLibrary("native-lib")

}

external fun apiKey1(): String

external fun apiKey2(): String

}

Once you've done this, you can then get a reference to your method by declaring an external Kotlin function under the same name as the one mentioned above in Step 3. In this example, it's apiKey().

Now, to get your API key from any part of your app, just call:

Keys.apiKey1()

Keys.apiKey2()