Installation Guide

Command-line tool:

1. We recommend testing the command-line tool from linprog as the JDK is already installed and configured there; for convenience, you may run

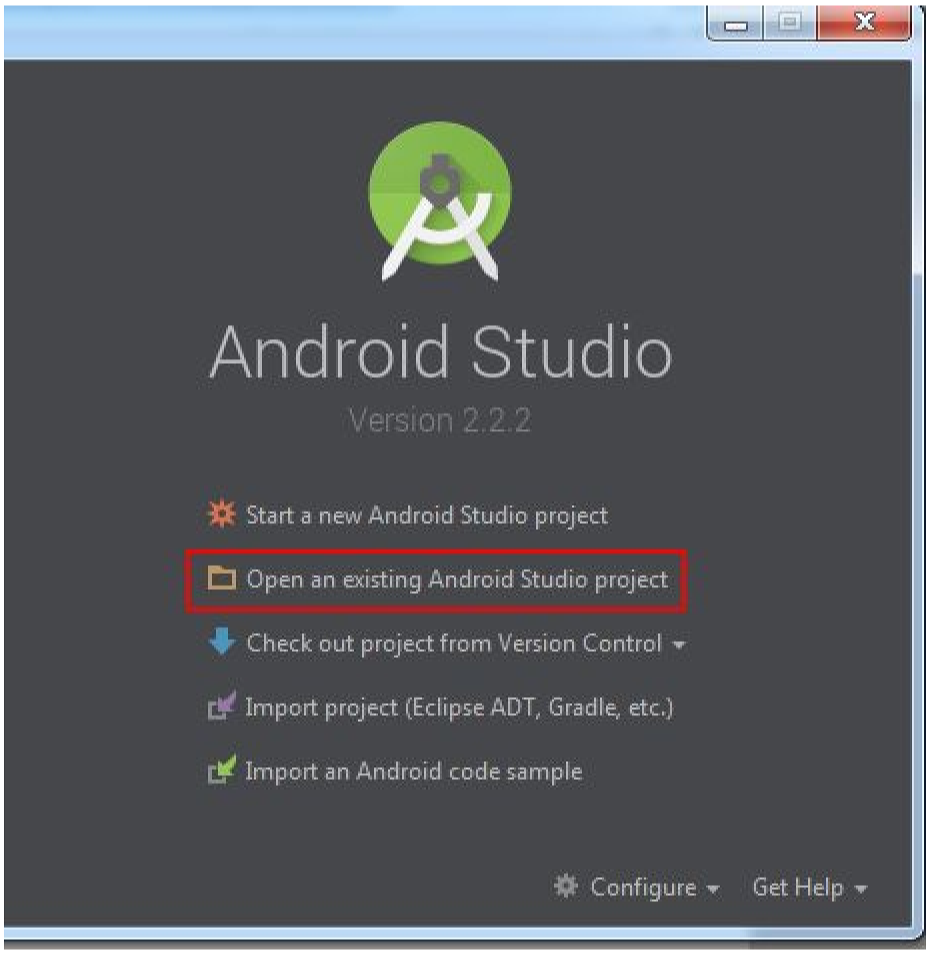
   `wget goo.gl/ZgoeKx`

   To fetch our project archive from GitHub from linprog. Note that you can unzip with the `unzip` command (which is also installed on linprog).
2. Run `make` to build the command-line tool.
3. Run `java lb_app` to run the command-line tool. The Functionality section covers the various commands that the app can perform.

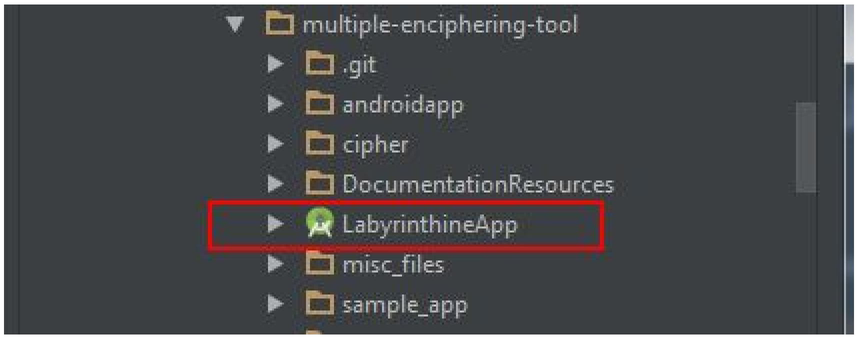
Android:

Note: the backend to the app is maintained by us - you will not need to install and configure any sort of backend yourself.

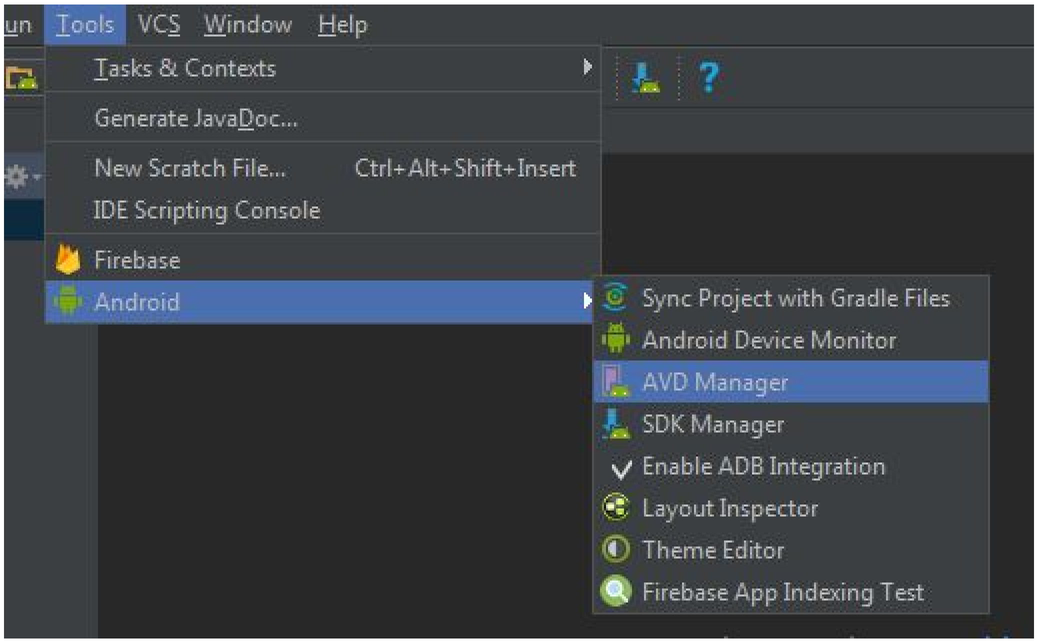
1. Make sure you have Android Studio installed (found here).
   a. If you have an Android device and the know-how, you may want to just install the provided Labyrinthine.apk instead.
2. Run Android Studio and open up an existing project:


Android Studio
Version 2.2.2
Start a new Android Studio project
Open an existing Android Studio project
Check out project from Version Control
Import project (Eclipse ADT, Gradle, etc.)
Import an Android code sample
Configure
Get Help

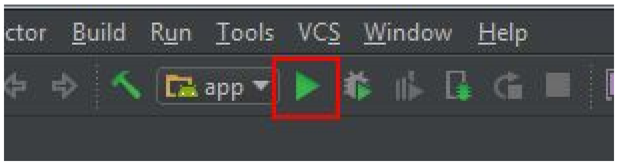
3. Navigate to the LabyrinthineApp project:



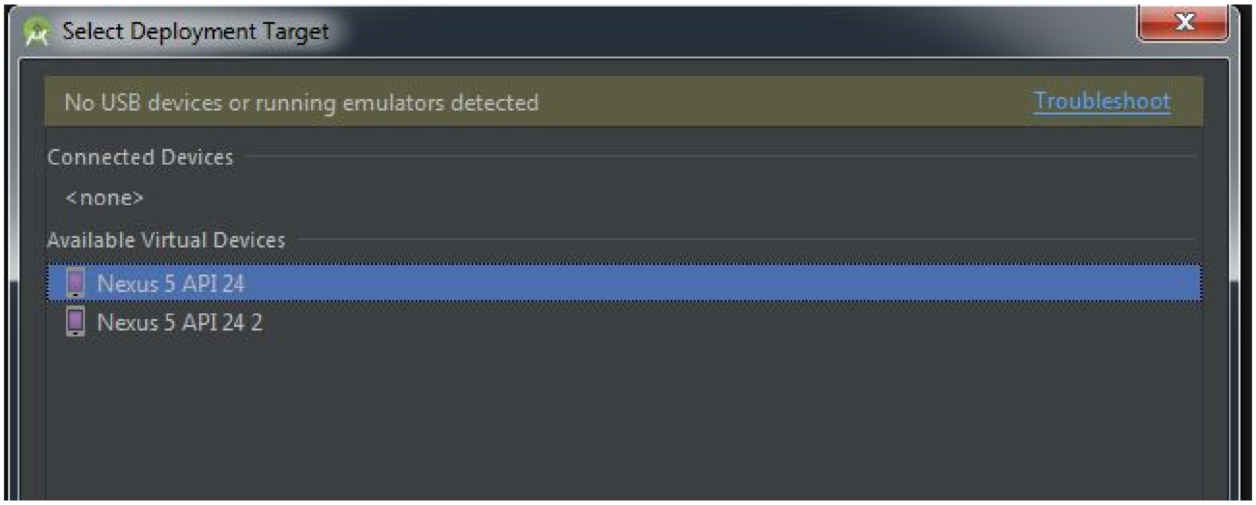
4. Next, you will need to create a virtual device to run the app on. Alternatively, if you have a physical Android device that you'd like to use, plug it in, enable USB Debugging from the Developer menu, and skip to step 6.
   To create the virtual device, navigate to the AVD Manager:



5. Create a new device - the default settings for category, device, system image, etc should be fine.
6. Build the app:



7. Choose your virtual/physical device from the list:



8. The app should compile and appear on the device all by itself.

Functionality

Command-line Tool:

- Text can be enciphered.
- Text can be deciphered.
- Input and output files can be specified.
- Specific ciphers can be specified.
  - The app can also read user-made ciphers, provided they are formatted properly.
- Standard in and standard out are used if no input and/or output file is given.
- The tool also implements a cipher library directory, by default created in the user's home directory. The location can be explicitly specified.

Android:

- The Labyrinthine app is a chat app that sends and receives encrypted messages.
- Accounts can be created within the app.
- New chats can be created, and the ciphers that they use can be changed.
- Other chats are visible, but the messages are enciphered.

Tutorial

Command-line Tool:

- Broadly speaking, lb_app takes text as an input, enciphers or decipers it, and outputs it. There are various flags that can be used to modify the behavior of the tool.
- The tool is invoked as so: `java lb_app`
- The tool *must* be given either `enc`, `encode`, `dec`, or `decode` as an argument.
- The tool must also be given a .cyph file: `-f <path to cyph file>`. There are several example files provided that you may use for testing in the cyph_files/ directory.

- The tool accepts `-i <input file path>` as an argument. This is the source text that will be enciphered or deciphered. If not given, the app will simply read from standard in.
- The tool accepts `-o <output file path>` as an argument. This is the destination for the enciphered/deciphered file. If not given, the app will simply output to standard out.
- The tool may use a library directory for storing ciphers as well. This allows you to simply specify the name of a .cyph file that's stored in the library instead of a path; this can be done with `-F <name of library cipher>`. If `-F` is used without `-L`, the tool will check ~/Labyrinthine/ for the requested .cyph name.
- The location of the cipher library can be specified with `-L <path to directory>`.

Here are some example commands (the given paths assume that you are at the root of the project folder):

- `java lb_app encode -i unciphered.txt -f cyph_files/orange.cyph`
- `java lb_app enc -i unciphered.txt -o ciphered.txt -f cyph_files/garnett.cyph`
- `java lb_app -i ciphered.txt -F garnett -L cyph_files/ decode`

There are also JUnit tests for the command-line tool. These can be run as follows:

- `make testprep`
- `make buildtest`
- `make runtest`

Android:

- The Labyrinthine app is a chat client built around the idea of sending and receiving enciphered messages. The app only supports one-on-one messaging at the moment.
- When the app is first launched, you will be prompted to either log in or register. You don't have to worry about using a legitimate email, the app does not require email verification. We don't recommend using one of your real emails or passwords.
- To start a new chat, open up the navigation drawer on the left side of the screen and press "New Chat."
    - You will be prompted to name the chat, provide your chat partner's email, and choose a cipher sequence to apply to the messages in the chat.
- The drawer also shows other chats. Currently, every chat that has been made with the app is visible. However, the chats will not be deciphered if you are not one of the members, and you cannot post messages to them.
- You can log out of the app via the options menu in the upper right.

Bad Inputs

Command-line Tool:

- The tool only works properly with ASCII input.
- The BookCipher and FourSquareCipher are not fully functional, so cipher sequences that include these will not work.

Android:

- The chat probably cannot handle a massive amount of new chats and/or chat spam. Please don't stress-test our app!
- Ciphers that use the BookCipher do not handle special characters (!/\#% etc) inside of messages to encipher and decipher.