VivoKey Chip Scan Library
Demo Code for Android

Introduction

The chip scan library (CSL) makes interacting with the VivoKey Scan API and extended APIs like the Key Value Store API easy. It handles communication between VivoKey APIs and the VivoKey chip implant through NFC. The demonstration app is meant to function as an introduction to using the chip scan library in your own mobile projects.

CSL Javadoc

A navigable Javadoc for the chip scan library can be found at;

https://api2.vivokey.com/docs/v1/chipscanlib/index.html

Demo App Source Code

To make learning how to use the CSL for Android easier, we have published source code for a demo app on GitHub. To use this demo source code, you will need to;

- Create an account or sign-in to https://developer.vivokey.com
- Create a new API key for testing purposes
- Have Android Studio and Git installed and configured
- Pull the demo app code from https://github.com/vivokey/chipscanlib-demo-android.git
- Download the chipscan library chipscanlib.jar file from http://api2.vivokey.com/lib/v1/chipscanlib-1.1-RELEASE.jar
- Import chipscanlib into the project
- Insert API key value for private static string API_KEY in MainActivity.java

Gradle config

You'll notice the dependencies section includes chipscanlib and all other necessary dependencies.

```
dependencies {
    implementation fileTree(dir: "libs", include: ["*.jar"])
    implementation 'androidx.appcompat:appcompat:1.2.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.0.4'
    implementation files('libs\\chipscanlib-1.1-RELEASE.jar')

    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'androidx.test.ext:junit:1.1.2'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.3.0'

}
```

Main activity

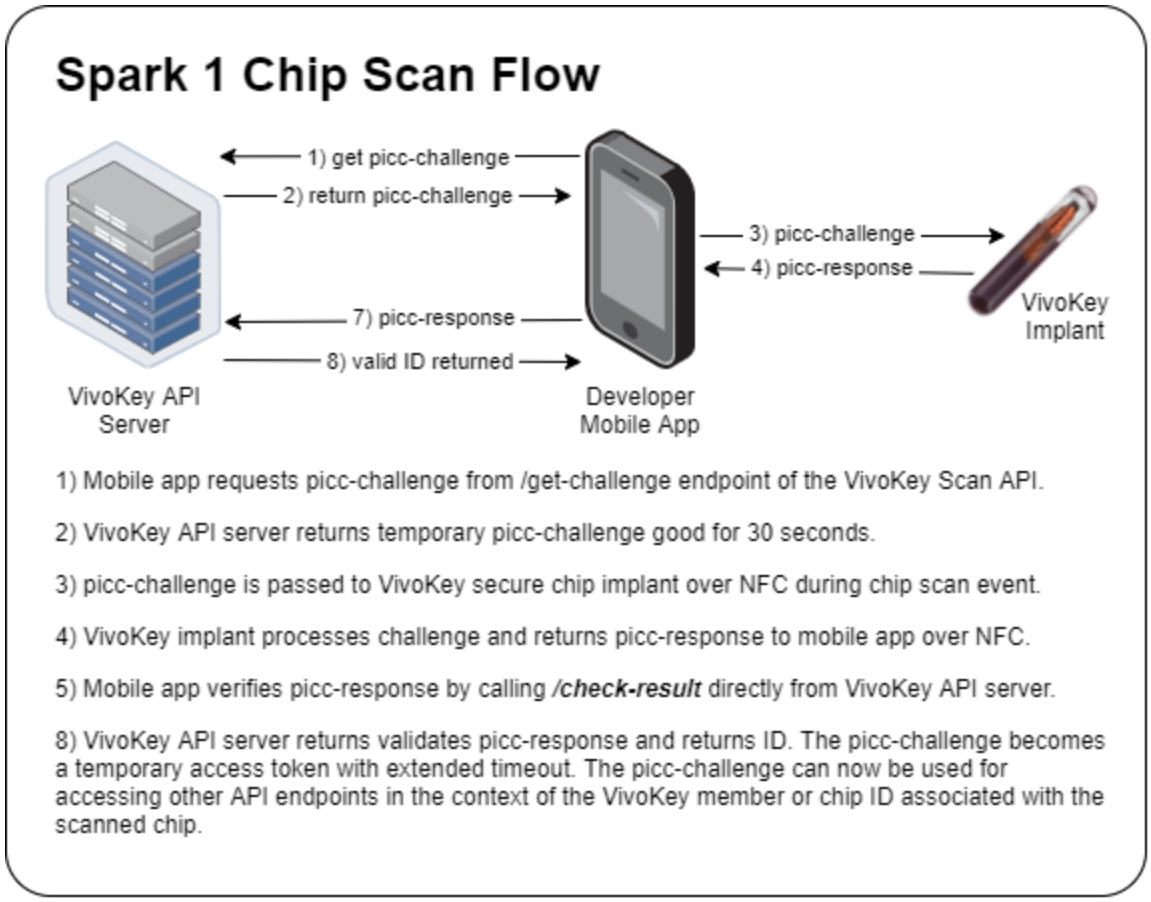
The bare minimum is NfcAdapter, Handler (for our implementation - if you have a different way to poll the NFC thread, that's fine), and Tag. Insert your VivoKey API key into the private static string API_KEY and you should be able to compile and run the demo app.

```
public class MainActivity extends AppCompatActivity {
    /**
     * These are the main variables required
     */

    // Set the API key here
    private static String API_KEY = "";
    // Represents the phone's NFC adapter
    private NfcAdapter mNfcAdapter;
    // Represents a Handler - gets used to poll the Chipscan library to see when it's finished
    private Handler mHandler;
    // Represents the Tag object received on an Intent with a NFC event in it
    private Tag mCurrentTag;
    // Becomes the authResult from the library, once authentication is complete
```

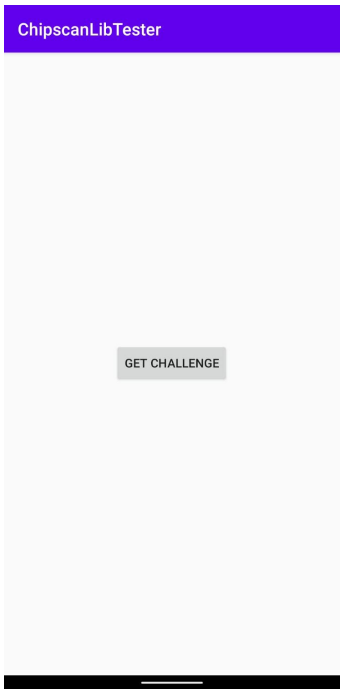
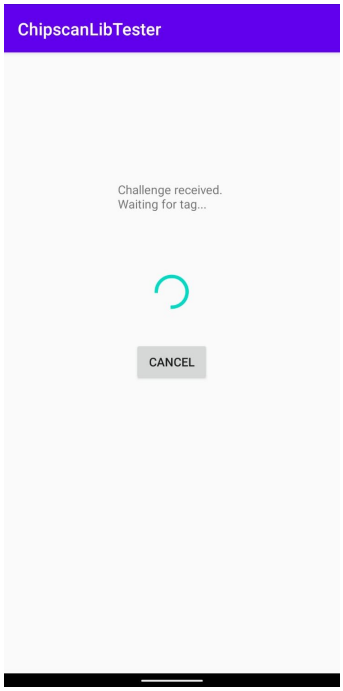
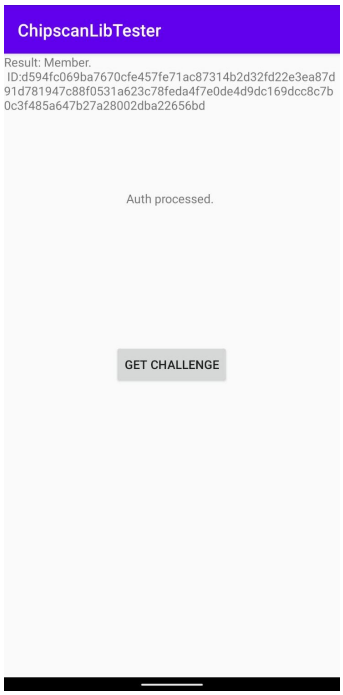
Chip Scan Event Flow

The basic process by which a VivoKey secure chip implant is cryptographically validated as authentic has slightly different flows depending on the VivoKey product being scanned, however the basic flow works the same way for all products. The first step is to obtain a challenge from the API server. This is processed by the chip implant and the result is returned to the server for confirmation. Here is a diagram of the most basic flow for a VivoKey Spark 1 chip.



Demo App Operation

The demo app demonstrates how to use the chip scan library to obtain a cryptographic challenge from the VivoKey API, initiate the NFC scanning process to present that challenge to the chip for processing, return the result to the VivoKey API, and obtain a usable ID. The library handles all communication between the chip and the VivoKey API, and in the end you will be able to obtain a member ID for the active VivoKey Member profile associated with that chip implant, or if you are using a properly licensed API key, obtain an authenticated chip ID for the VivoKey product scanned, which is separate from the VivoKey identity platform's member ID.

Open app	Get challenge, start scanning	Scan chip, get ID
		

1) Tapping the “GET CHALLENGE” button starts the process of grabbing a fresh new cryptographic challenge from the VivoKey API server. It is important to be able to get a challenge before you prompt the user to scan their chip because calls to web APIs can sometimes take a lot of time and can be slowed significantly by poor signal strength, network latency, or overall congestion. By getting the challenge first, we spare the person from having to hold their chip to the phone for any more time than absolutely necessary. Also, you are able to confirm connectivity with the VivoKey API server before prompting the person to scan their chip.

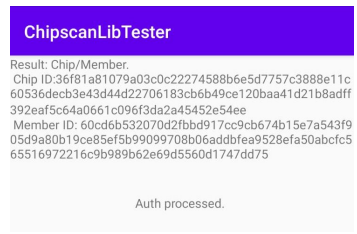
2) Once a challenge has been received, we can prompt the user to scan their chip and initiate chip scanning via the library. Communication is established over NFC between your app and the chip (via the library), and the challenge is given to the chip for processing.

3) The cryptographic response is returned from the chip to the app over NFC, and the library returns that to the VivoKey API. If the response is correct for the given cryptographic challenge, one or two IDs are returned.

Types of IDs returned

There are two types of IDs which can be returned upon a successful VivoKey chip scan. The IDs returned are a unique hash of your internal developer account ID, a large random salt, and unique data pertaining to the VivoKey member or the VivoKey product which was scanned. This means that the ID received for a given member or product will be unique and identical for all API keys generated with your developer account. Any API keys generated by any other developer account will get a different unique ID for the member or product.

member-id	A member ID is associated with the VivoKey member, not a specific VivoKey chip or product. If the member has multiple VivoKey products associated with their profile, any of those products will produce the same member ID when scanned. ***Products which are new, unclaimed, and not associated with any VivoKey member will not return any member ID, and instead an error code will be returned.***
chip-id	A chip ID is an authenticated ID associated only with the exact chip or VivoKey product which was scanned. It is entirely disassociated the VivoKey identity platform, meaning a chip ID will always be returned for a successful authentication regardless of whether or not that chip has been associated with a VivoKey member profile or not. ***Access to chip ID is not enabled by default. Chip ID access requires additional licensing, and can be enabled on a per API key basis.*** If a VivoKey chip or product is scanned using an API key with chip ID enabled, and the chip or product is also associated with a VivoKey member profile, you will receive both types of IDs upon successful scan.



Member and Chip ID both reported

In a situation where you have a fully licensed API key with chip ID enabled, and your application scans a VivoKey product which has been successfully associated with a VivoKey member profile, you will receive both a member ID and a chip ID. Your application will need to decide how to use these IDs.