Keeping Secrets Secure

Fingerprint Authentication & the Android Keystore

Josh Skeen | josh@bignerdranch.com | @mutexkid

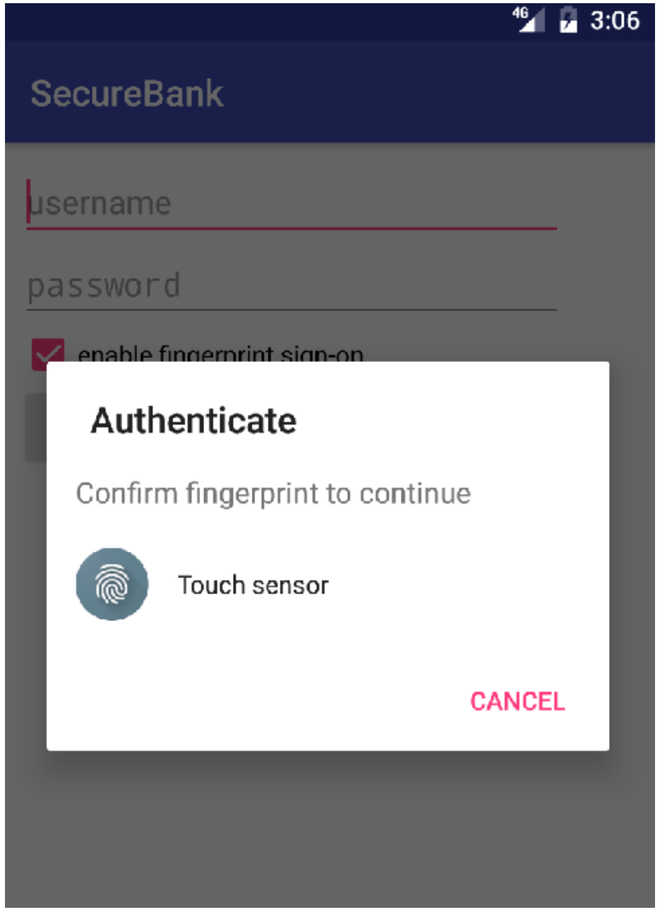
Who am I?

Android Developer & Instructor at Big Nerd Ranch

Josh Skeen | josh@bignerdranch.com | @mutexkid

Big Nerd Ranch

SecureBank



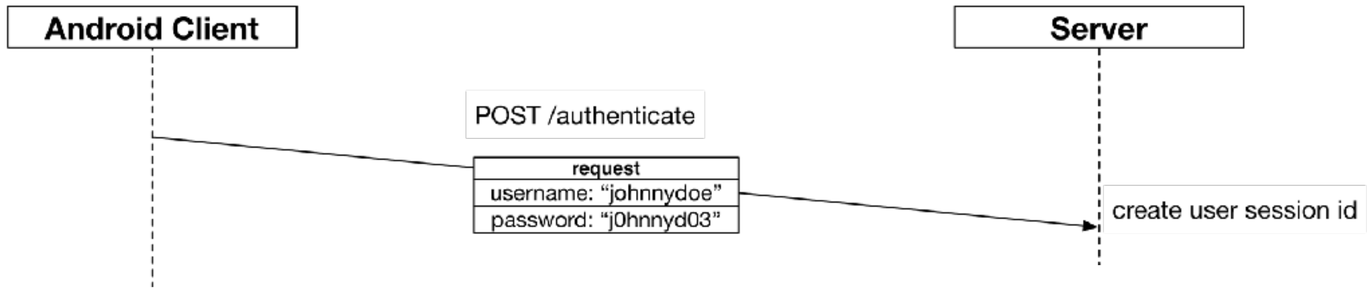
Big Nerd Ranch

http://github.com/bignerdranch/android-securebank

The Story So Far

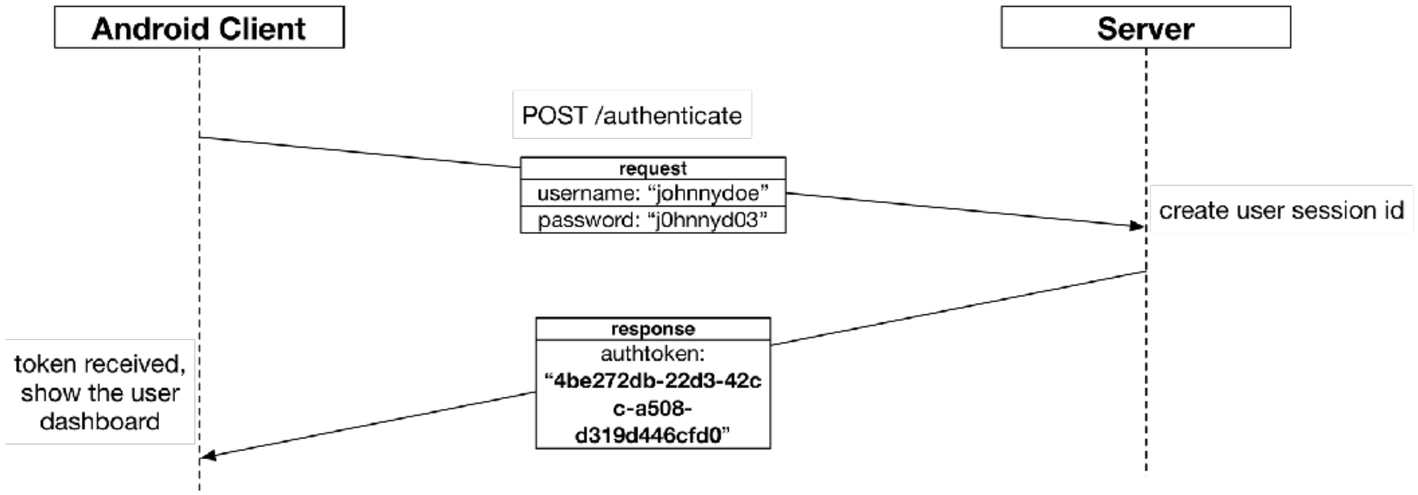
(typical web service situation)

The Story So Far



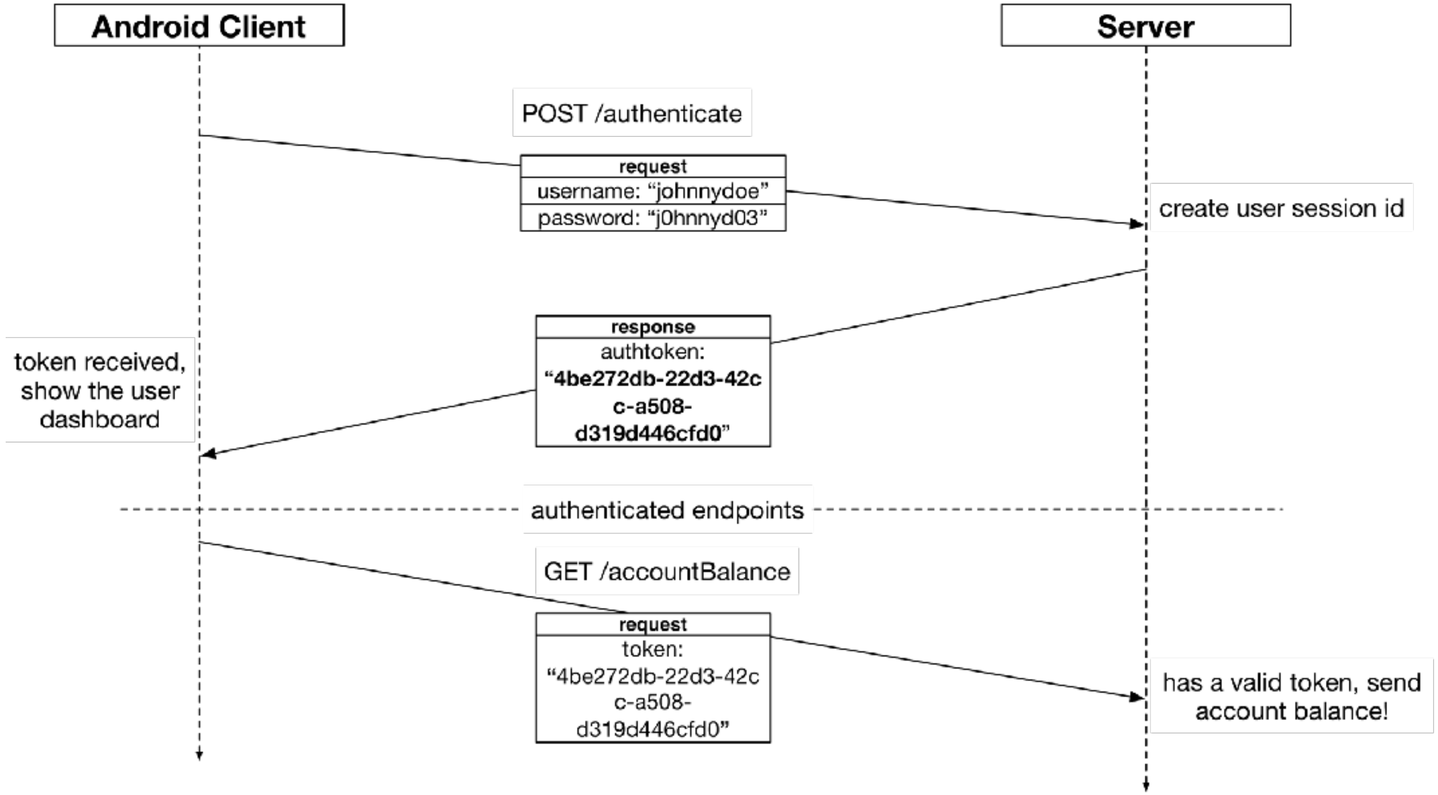
(typical web service situation)

The Story So Far



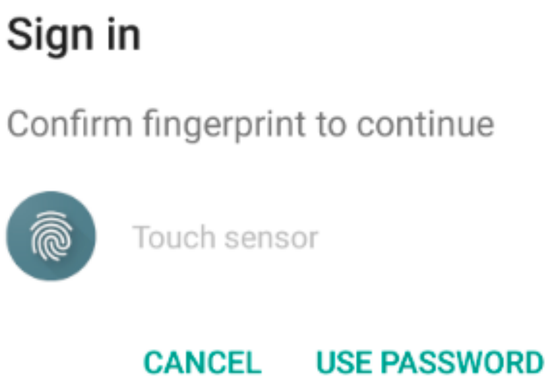
(typical web service situation)

The Story So Far



(typical web service situation)

New Feature



A user should be allowed to sign in using their fingerprint, optionally.

Central Problem

What should i save locally if i want to implement fingerprint authentication, and how can i do it securely ?

Getting it Wrong!

Storing the **password itself** in the application Shared Preferences

/data/data/securebankcorp.mobile.client.android/shared_prefs:

```
<map>
  <string name="username">johnnydoe</string>
  <string name="password">pizza2017</string>
  <boolean name="rememberUsername" value="true" />
</map>
```

Getting it Wrong!

Encrypting the **password or token**, but **using keys and encryption constants that are defined in application code**


```
<map>
   <string name="username">johnnydoe</string>
      <string
name="password">2A52D2D7A0F3802DB9E7AE118C99F3F1B04C07240601C0EFC8EAC0E28E22198
17B2FEC7376BD53B7D606288219D7E03AAD5A899D706BC1448D44B1D908A2DD416480652CF29F
1EEAF20AA93C4224991218E058B1C4BA4873E6363F2F0E5A35A773B21B78541EA54BFFB3D22AAE2
39A0CDD20CC50F8E6201F7B6A4D7B4AB73C46A92B0109E559343CDE6B170B1B6DAE63B42EEA8DD
CCEBFC84FD610F9F9FEDFE8D8432843FEF65954D1F67919FA06AEFABF1AFEEA23B071DA478851B5E
1973664D81104B845D899CCFF437E390C40CD2D5E3038C87C4FCBD8B9F49774FB54CDC0</
string>
</map>
```


Big Nerd Ranch

Getting it Wrong!

Encrypting the **password or token**, but **using keys and encryption constants that are defined in application code:**

```
.class public Lcom/securebanking/MainActivity;
.super Landroid/app/Activity;
.source "MainActivity.java"

# static fields
.field static final synthetic $assertionsDisabled:Z

.field static final DEFAULT_KEY_NAME:Ljava/lang/String; = "default_key"

.field private static final SECRET_KEY:Ljava/lang/String; = "NOTSOSECRET"

.field private static final TAG:Ljava/lang/String;

```

apktool -d release.apk

Proposed Solution

- Upon a user authenticating, capture the session token
- Create a SecretKey and store it in the **AndroidKeyStore**
- Use a Cipher w/ SecretKey to encrypt & decrypt the token
- Hook it up to work with the Fingerprint API

code at :

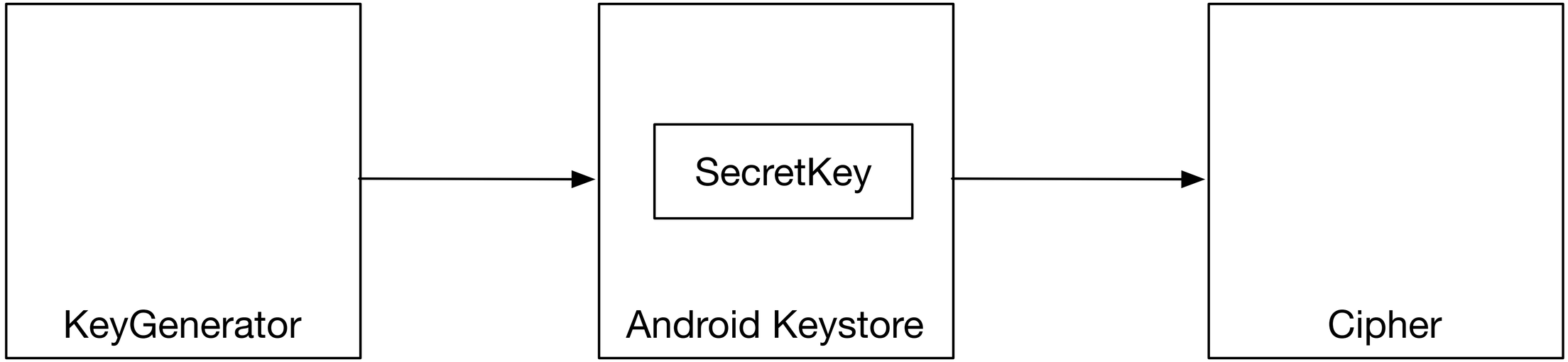
http://github.com/bignerdranch/android-securebank

also check out:

https://github.com/googlesamples/android-FingerprintDialog

Big Nerd Ranch

Creating A Secret Key



Make a (secret) Key

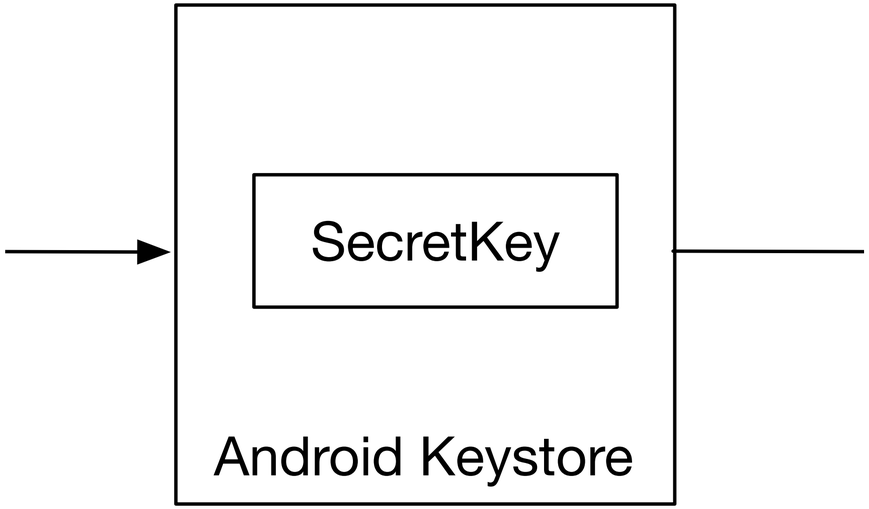
Save the Key (securely!)

Encrypt the Data

KeyStore

…stores keys (and certificates)

AndroidKeyStore



Save the Key (securely!)

- Can store Public Keys, Private Keys, and Certificates
- In our case, all we'll need is a Private Key (symmetric)
- Does Fancy stuff under the hood (TEE, SE, etc)
- Actually a "KeystoreProvider"
- Been called other things in previous versions

Prep the Keystore

```
private static final String ANDROID_KEY_STORE =
"AndroidKeyStore";
```

```
keystore = KeyStore.getInstance(ANDROID_KEY_STORE);
keystore.load(null);
```

KeyStoreHelper.java

Next Up: Creating a Secret Key

```
KeyGenerator keyGenerator = KeyGenerator
    .getInstance(KeyProperties.KEY_ALGORITHM_AES, ANDROID_KEY_STORE);
```

KeyStoreHelper.java

KeyGenParameterSpec

Provides Config for how specifically we want to construct the SecretKey

```
KeyGenParameterSpec.Builder builder =
    new KeyGenParameterSpec.Builder(DEFAULT_KEY,
    KeyProperties.PURPOSE_ENCRYPT | KeyProperties.PURPOSE_DECRYPT)
   .setBlockModes(KeyProperties.BLOCK_MODE_CBC)
   .setEncryptionPaddings(KeyProperties.ENCRYPTION_PADDING_PKCS7);
```

KeyStoreHelper.java

Generate the Key

```
keyGenerator.init(builder.build());
keyGenerator.generateKey();
```

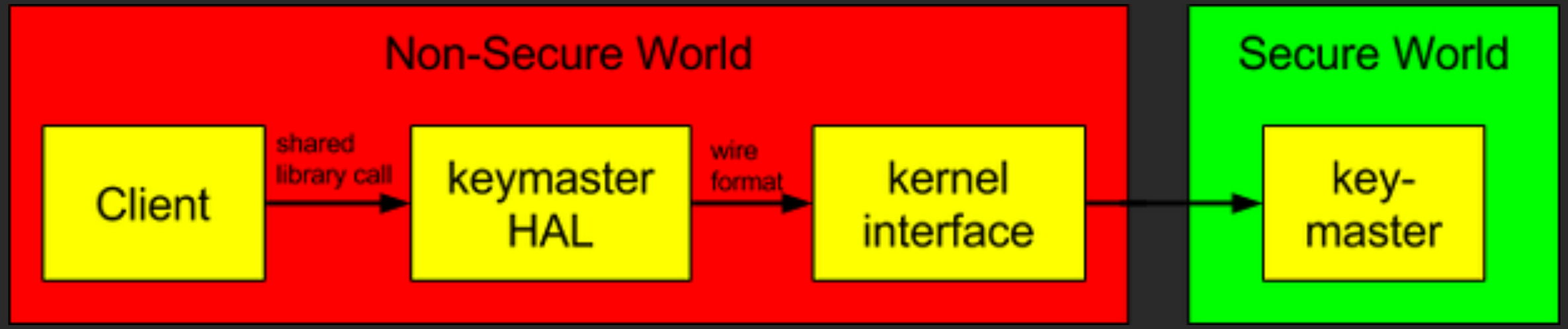
KeyStoreHelper.java

Under the Hood…

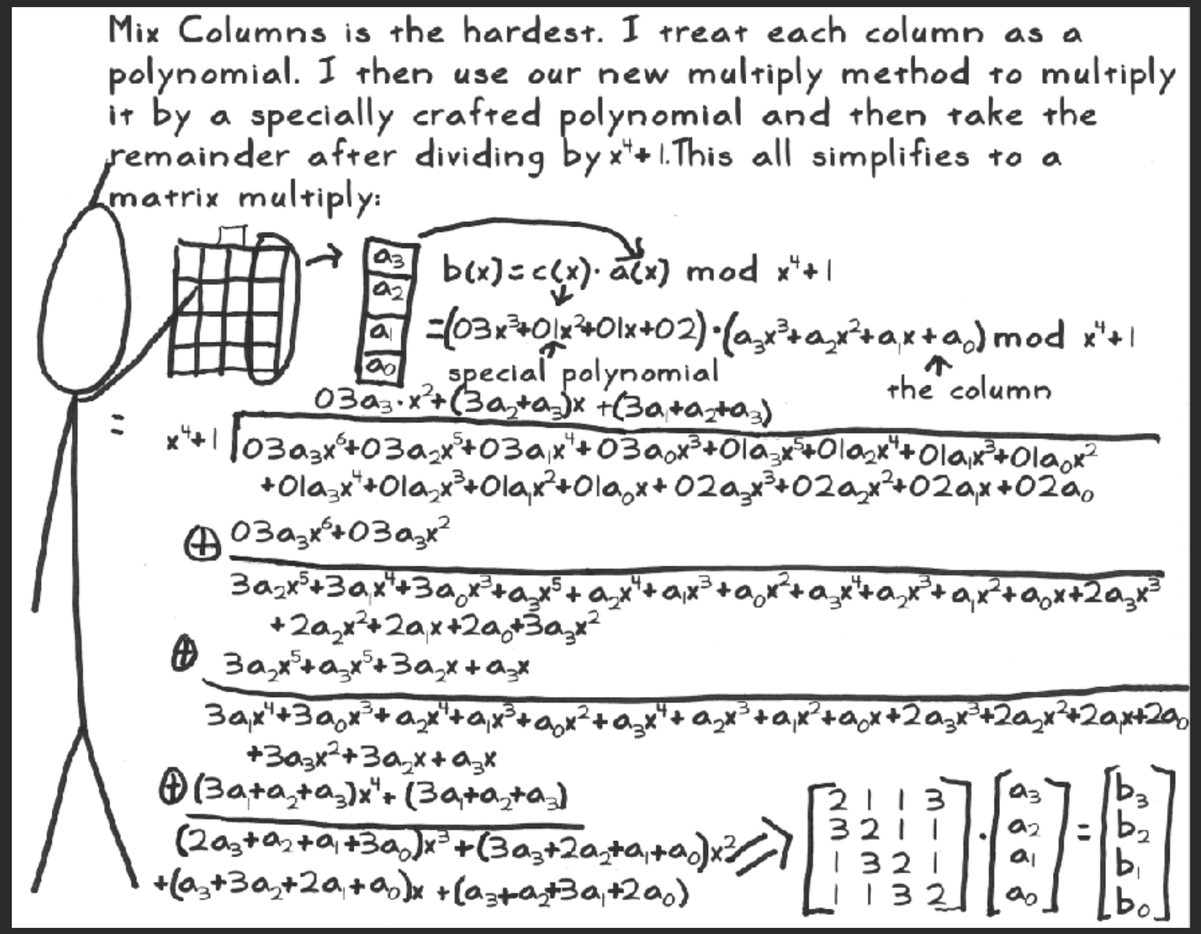
```
joshskeen@joshs-MacBook-Pro-3.local:/Users/joshskeen/Desktop/keep-secrets-secure  git:(master*) $ adb shell
generic_x86:/ $ su
generic_x86:/ # cd /data/misc/keystore/user_0
generic_x86:/data/misc/keystore/user_0 # ls
10109_USRSKEY_default_key
generic_x86:/data/misc/keystore/user_0 #
```

Big Nerd Ranch

Under the Hood...



AES??



Big Nerd Ranch

http://www.moserware.com/2009/09/stick-figure-guide-to-advanced.html

Next Up: Encrypting the Token

Cipher needs a SecretKey

Obtain a Cipher Instance

```
Cipher cipher =
Cipher.getInstance(KeyProperties.KEY_ALGORITHM_AES + "/"
    + KeyProperties.BLOCK_MODE_CBC + "/"
    + KeyProperties.ENCRYPTION_PADDING_PKCS7);
```

CryptoHelper.java

Initializing the Cipher

```
private void initEncryptCipher(Cipher cipher) {
    cipher.init(Cipher.ENCRYPT_MODE,
                keyStoreHelper.getSecretKey());
    sharedPreferencesHelper.saveIV(cipher.getIV());
}
```

CryptoHelper.java

Storing the IV (needed later!)

```
private void initEncryptCipher(Cipher cipher) {
    cipher.init(Cipher.ENCRYPT_MODE,
                keyStoreHelper.getSecretKey());
    sharedPreferencesHelper.saveIV(cipher.getIV());
}
```

CryptoHelper.java

Putting It All Together: doFinal()

```
public void encrypt(String authTokenFromServer) {
    Cipher cipher = getCipher();
    initEncryptCipher(cipher);
    byte[] bytes =
        cipher.doFinal(authTokenFromServer.getBytes());
    sharedPreferencesHelper.saveToken(bytes);
}
```

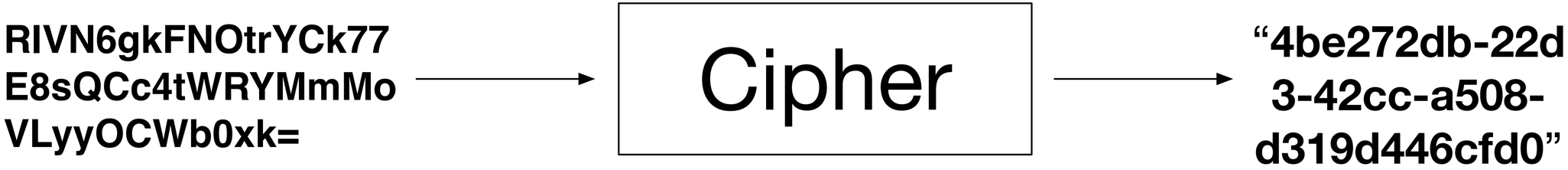
CryptoHelper.java

Saving the Encrypted Token

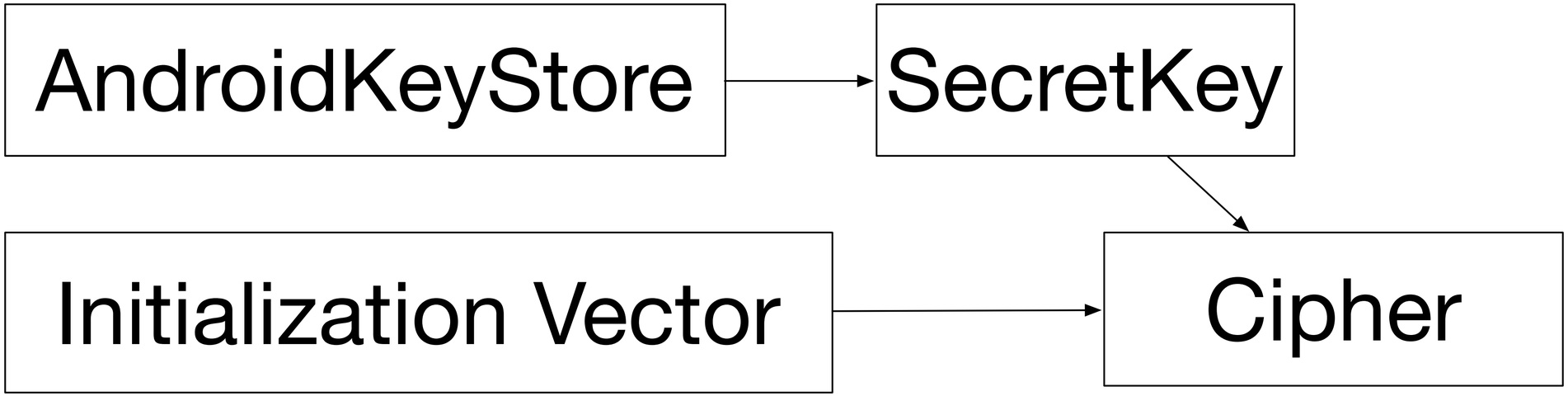
```
public void saveToken(byte[] encryptedToken) {
    String encoded = Base64.encodeString(encryptedToken, Base64.NO_WRAP);
    sharedPreferences
    .edit()
    .putString(SHARED_PREFERENCES_ENCRYPTED_TOKEN, encoded).apply();
}
```

SharedPreferencesHelper.java

Next: Decrypting the Token



Cipher for Decrypt



Decrypt: Retrieve IV

```
byte[] iv = sharedPreferencesHelper.getIV();
IvParameterSpec ivParameterSpec = new IvParameterSpec(iv);
```

CryptoHelper.java

Decrypt: IVParameterSpec

```
byte[] iv = sharedPreferencesHelper.getIV();
IvParameterSpec ivParameterSpec = new IvParameterSpec(iv);
```

CryptoHelper.java

Decrypt: Initialize the Cipher

```
cipher.init(Cipher.DECRYPT_MODE,
            keyStoreHelper.getSecretKey(),
            ivParameterSpec);
```

CryptoHelper.java

Decrypting the Token

```
String textToDecrypt =
    sharedPreferencesHelper.getEncryptedToken();
```

CryptoHelper.java

Decrypting the Token

```
ByteArrayInputStream is =
    new ByteArrayInputStream(Base64.decode(textToDecrypt,
                                           Base64.DEFAULT));
```

CryptoHelper.java

Problem 2: Fingerprint Authentication

FingerprintManager – system service for managing fingerprint reader hardware/software interaction

CryptoObject – verifies the fingerprint authentication occurred and enables or revokes a Cipher object in ENCRYPT or DECRYPT mode

Fingerprint Sensor Technology

- most are using capacitive touch
- old tech = optical – sort of dangerous!
- new tech = ultrasonic

Adding USE_FINGERPRINT Permission

```
<uses-permission android:name="android.permission.USE_FINGERPRINT"/>
```

AndroidManifest.xml

Adding USE_FINGERPRINT Permission

```
requestPermissions(new String[]{Manifest.permission.USE_FINGERPRINT},
                   REQUEST_USE_FINGERPRINT);

@Override
public void onRequestPermissionsResult(int requestCode,
    @NonNull String[] permissions,
    @NonNull int[] grantResults) {
if (requestCode == REQUEST_USE_FINGERPRINT &&
    grantResults[0] == PackageManager.PERMISSION_GRANTED) {
            performFingerprintAuthentication();
        }
    }
}
```

LoginActivity.java

Fingerprint Auth: Check if Available

```
private void performFingerprintAuthentication() {
    if (fingerprintManager.isHardwareDetected() &&
        fingerprintManager.hasEnrolledFingerprints()) {
      //ready to rock. warm up the sensor
    } else {
        //no dice. show an error message, etc
        Timber.e("fingerprint not available");
    }
}
```

LoginActivity.java

Big Nerd Ranch

KGPS: Enabling User Auth Required

```
KeyGenParameterSpec.Builder builder = new KeyGenParameterSpec
    .Builder(DEFAULT_KEY, KeyProperties.PURPOSE_ENCRYPT |
    KeyProperties.PURPOSE_DECRYPT)
    .setBlockModes(KeyProperties.BLOCK_MODE_CBC)
    .setEncryptionPaddings(KeyProperties.ENCRYPTION_PADDING_PKCS7)
    .setUserAuthenticationRequired(true);
```

throws an android.security.KeyStoreException if used w/ cipher and user’s not “authed"

KeyStoreHelper.java

Big Nerd Ranch

Fingerprint Auth: CryptoObject

```
Cipher cipher = cryptoHelper.getCipher();
cryptoHelper.initDecryptCipher(cipher);

FingerprintManager.CryptoObject cryptoObject =
    new FingerprintManager.CryptoObject(cipher);
```

LoginActivity.java

Fingerprint Manager : Authenticate

```
fingerprintManager.authenticate(cryptoObject, authCallback);
```

LoginActivity.java

Fingerprint Authentication Callbacks

```
private void performFingerprintAuthentication() {
    Cipher cipher = cryptoHelper.getCipher();
    cryptoHelper.initDecryptCipher(cipher);
    FingerprintManager.CryptoObject cryptoObject = new FingerprintManager.CryptoObject(cipher);
    fingerprintManager.authenticate(cryptoObject, null, 0, new FingerprintManager.AuthenticationCallback() {
        @Override
        public void onAuthenticationFailed() {
            super.onAuthenticationFailed();
        }
        @Override
        public void onAuthenticationSucceeded(FingerprintManager.AuthenticationResult result) {
            super.onAuthenticationSucceeded(result);
            //we can now decrypt using our authenticated cipher!
        }
        @Override
        public void onAuthenticationError(int errorCode, CharSequence errString) {
            super.onAuthenticationError(errorCode, errString);
        }
    }, null);
}
```

LoginActivity.java

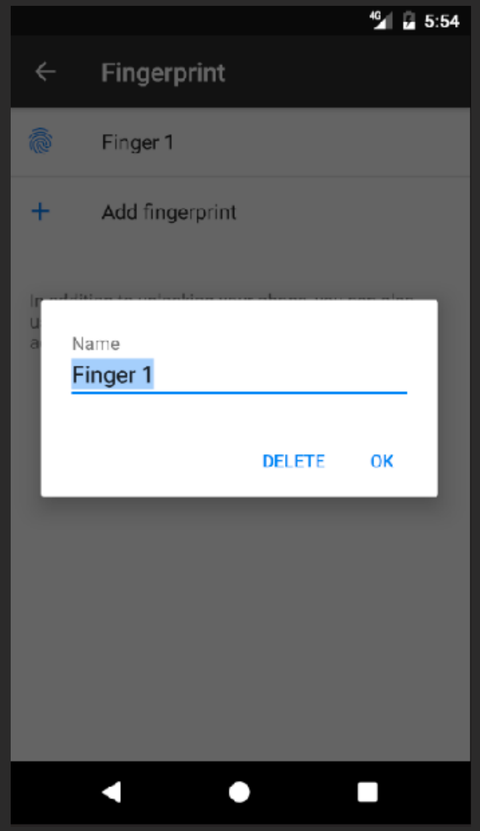
Big Nerd Ranch

Decrypt, using the Authed Cipher

```
@Override
public void onAuthenticationSucceeded(AuthenticationResult result) {
    super.onAuthenticationSucceeded(result);
    String decryptedToken = cryptoHelper.decrypt(result.getCryptoObject());
    //we can now decrypt using our authenticated cipher
}
```

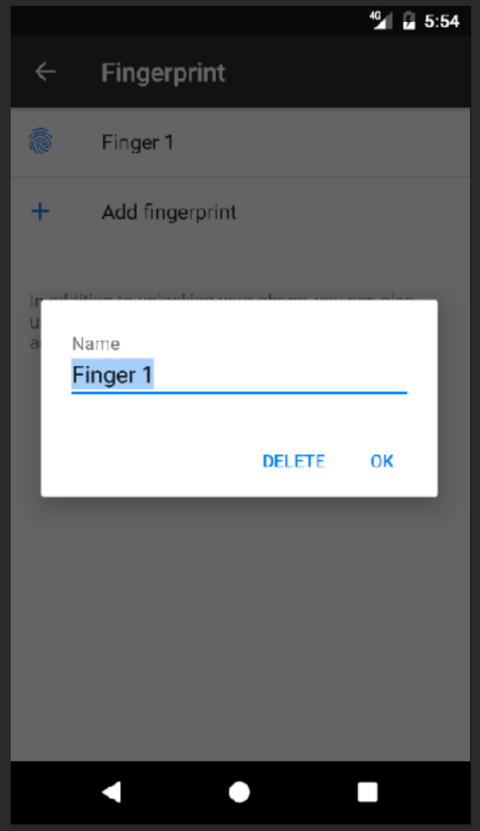
LoginActivity.java

Gotchas : Finger Deleted



?

Gotchas : Finger Deleted



KeyPermanentlyInvalidatedException!

Big Nerd Ranch

Further Keystore-Related Tricks

KeyInfo:

```
.isUserAuthenticationRequirementEnforcedBySecureHardware()
.isKeyInsideSecureHardware()
```

SecretKeyGenerator:

```
setUserAuthenticationValidityDurationSeconds()
```

Note about Testing

- robolectric “shadow object” doesn't work with keystore currently
- real integration tests (espresso) may be best option if its important

Final Thoughts

- Good foundation, though with security there's never a "one size fits all" solution
- Additional measures available – Proguard, Dexguard, SafetyNet (is it "security theater?")
- 2FA/MFA
- Big Nerd Ranch Security Courses coming soon! (Android & Web) Q3ish 2017

Resources

- code example: https://github.com/bignerdranch/securebank
- SafetyNet API: https://developer.android.com/training/safetynet/index.html
- Big Nerd Ranch Security Course: http://www.bignerdranch.com
- KeyGenerator : https://developer.android.com/reference/javax/crypto/KeyGenerator.html
- AES explained with stick figures: www.moserware.com/2009/09/stick-figure-guide-to-advanced.html
- IDA https://www.hex-rays.com/products/ida/
- FRIDA https://www.frida.re/docs/android/

Big Nerd Ranch

Thanks!

Josh Skeen | josh@bignerdranch.com | @mutexkid

https://github.com/bignerdranch/stockwatcher

Josh Skeen | josh@bignerdranch.com | @mutexkid

Big Nerd Ranch

Big Nerd Ranch