OpenPGP API: A Case Study of AIDL Versioning

Android Stammtisch January 2015

Dominik Schürmann

2015-01-28

$ whois

Dominik Schürmann

- Employed at Technische Universität Braunschweig
- Network security research
- OpenPGP: Open standard for end-to-end email security
- Hobbyist Android developer
  - AdAway
  - F-Droid contributions
  - K-9 contributions
  - OpenKeychain: OpenPGP on Android

Activity Intents

- Intents with Bundles
- "An Intent is a messaging object you can use to request an action from another app component."

```
Intent intent = new Intent(getActivity(), PassphraseDialogActivity.class);
intent.putExtra(PassphraseDialogActivity.EXTRA_SUBKEY_ID, mSignMasterKeyId);
startActivityForResult(intent, REQUEST_CODE_PASSPHRASE);
```

```
@Override
public void onActivityResult(int requestCode, int resultCode, Intent data) {
    switch (requestCode) {
        case REQUEST_CODE_PASSPHRASE: {
            if (resultCode == Activity.RESULT_OK && data != null) {
                String passphrase = data.getStringExtra(
                    PassphraseDialogActivity.MESSAGE_DATA_PASSPHRASE);
            }
            return;
        }
        default: {
            super.onActivityResult(requestCode, resultCode, data);
        }
    }
}
```

Broadcast Intents and BroadcastReceivers

- Send broadcast Intent from anywhere to BroadcastReceiver
- “A broadcast is a message that any app can receive.”
- Ordered Broadcasts
- Local Broadcasts

```
Intent i = new Intent("org.sufficientlysecure.keychain.USER_ACTION");
sendBroadcast(i);
```

```
public class MyReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
    }
}
```

Service Intents

- Start service from anywhere with Intent

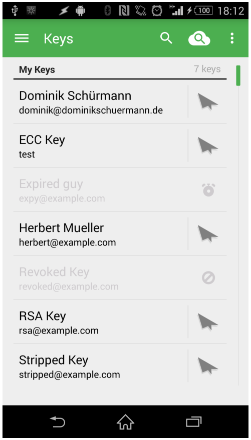
```
Intent intent = new Intent(this, KeychainIntentService.class);
intent.setAction(KeychainIntentService.ACTION_ENCRYPT);

// Create a new Messenger for the communication back
Messenger messenger = new Messenger(saveHandler);
intent.putExtra(KeychainIntentService.EXTRA_MESSENGER, messenger);

// start service with intent
startService(intent);
```

Use Case/Requirements

OpenKeychain from git, branch development / K-9 Mail


18:12

Keys

My Keys 7 keys

Dominik Schürmann
dominik@dominikschuermann.de

ECC Key
test

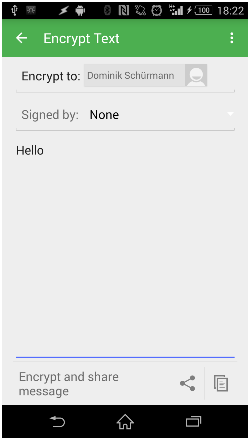
Expired guy
expy@example.com

Herbert Mueller
herbert@example.com

Revoked Key
revoked@example.com

RSA Key
rsa@example.com

Stripped Key
stripped@example.com


18:22

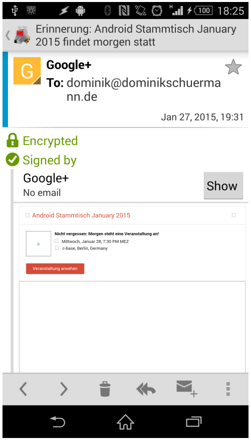
Encrypt Text

Encrypt to: Dominik Schürmann

Signed by: None

Hello

Encrypt and share message


18:25

Erinnerung: Android Stammtisch January 2015 findet morgen statt

Google+
To: dominik@dominikschuermann.de

Jan 27, 2015, 19:31

Encrypted

Signed by

Google+
No email

Show

Android Stammtisch January 2015

Nicht vergessen: Morgen steht eine Veranstaltung an!

Mittwoch, Januar 28, 7:30 PM MEZ

c-base, Berlin, Germany

Veranstaltung ansehen

Use Case/Requirements

- Expose API for other apps to encrypt/decrypt/sign/verify content
- Content can be quite large $\Rightarrow$ streams!
- Real Inter-process communication with **optional** user interaction
- API should be as easy as possible
- Some user interaction should be done by OpenKeychain, such as passphrase input

Use Case/Requirements

- Expose API for other apps to encrypt/decrypt/sign/verify content
- Content can be quite large $\Rightarrow$ streams!
- Real Inter-process communication with **optional** user interaction
- API should be as easy as possible
- Some user interaction should be done by OpenKeychain, such as passphrase input

This is madness!

Problems

- Activity Intents are easily exposed for other apps, but require user interaction
- Bundle extras are limited in size (˜1 MB)
- No streams over Broadcasts

```
ALOGE("!!! FAILED BINDER TRANSACTION !!!");
// TransactionTooLargeException is a checked exception, only throw from certain
   methods.
// FIXME: Transaction too large is the most common reason for FAILED_TRANSACTION
//      but it is not the only one.  The Binder driver can return BR_FAILED_REPLY
//      for other reasons also, such as if the transaction is malformed or
//      refers to an FD that has been closed.  We should change the driver
//      to enable us to distinguish these cases in the future.
```

Android Interfaces

- "It allows you to define the programming interface that both the client and service agree upon in order to communicate with each other using interprocess communication (IPC)"

```
// IRemoteService.aidl
package com.example.android;

// Declare any non-default types here with import statements

/** Example service interface */
interface IRemoteService {
    /** Request the process ID of this service, to do evil things with it. */
    int getPid();

    /** Demonstrates some basic types that you can use as parameters
     * and return values in AIDL.
     */
    void basicTypes(int anInt, long aLong, boolean aBoolean, float aFloat,
            double aDouble, String aString);
}
```

Implementation of Interface and Service

```
public class RemoteService extends Service {
    @Override
    public void onCreate() {
        super.onCreate();
    }
    @Override
    public IBinder onBind(Intent intent) {
        // Return the interface
        return mBinder;
    }
    private final IRemoteService.Stub mBinder = new IRemoteService.Stub() {
        public int getPid(){
            return Process.myPid();
        }
        public void basicTypes(int anInt, long aLong, boolean aBoolean,
            float aFloat, double aDouble, String aString) {
            // Does nothing
        }
    };
}
```

Connect

```java
IRemoteService mIRemoteService;
private ServiceConnection mConnection = new ServiceConnection() {
    // Called when the connection with the service is established
    public void onServiceConnected(ComponentName className, IBinder service) {
        // Following the example above for an AIDL interface,
        // this gets an instance of the IRemoteInterface, which we can use to
            call on the service
        mIRemoteService = IRemoteService.Stub.asInterface(service);
    }

    // Called when the connection with the service disconnects unexpectedly
    public void onServiceDisconnected(ComponentName className) {
        Log.e(TAG, "Service has unexpectedly disconnected");
        mIRemoteService = null;
    }
};
```

Problems

- Not easily versionable
- Backward compatibility problems: What if a new parameter or method is introduced
- How to pass objects?
- What about input/output streams?
- User Interaction if required, e.g., passphrase input?

Versionable and Backward Compatible Method Parameters

Use Intent/Bundle inside of AIDL method definition!

Demo Code

Input-/Outputstreams

Use ParcelFileDescriptors and stream in/out with them using pipes

Demo Code

Note: Not usable inside parameters Bundle!

Objects

- Use Parcelables
- Use versionable Parcelables (seen in Dashclock Widget)
- Put them in new parameters Bundle for passing them around

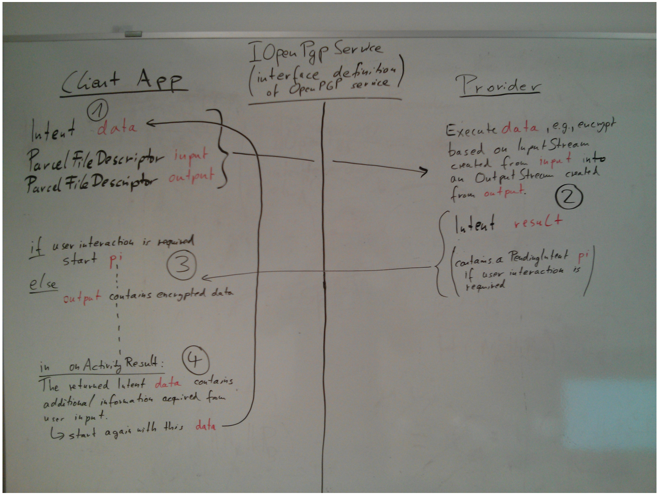
Demo Code

User Interaction

- A little bit more complicated
- Google's billing API uses something similar
- Return PendingIntent to client application with predefined set of extras
- Client application can execute this PendingIntent with `startIntentSenderForResult()`
- In `onActivityResult` restart process using newly returned parameter Bundle

Demo Code (`RemoteService`)

Process



Client App

IOpenPgpService
(interface definition of OpenPGP service)

Provider

①
Intent data
ParcelFileDescriptor input
ParcelFileDescriptor output

Execute data, e.g., encrypt based on InputStream created from input into an OutputStream created from output. ②

Intent result
(contains a PendingIntent pi if user interaction is required)

if user interaction is required
start pi ③
else
output contains encrypted data

in onActivityResult: ④
The returned Intent data contains additional information acquired from user input.
↳ start again with this data

Demo Code

Custom Permissions

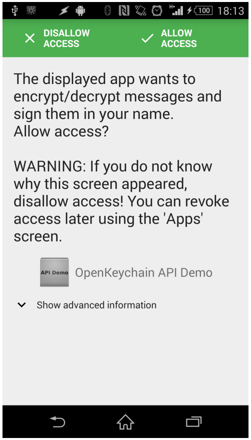
Restricting access to API with custom permissions?

- Extreme Limitations
- Attack scenarios
- `https://github.com/commonsguy/cwac-security/blob/master/PERMS.md`
- First one in wins. In other words, the first app (or framework, in the case of the OS's platform permissions) that defines a `<permission>` for a given android:name gets to determine what the description is and what the protection level is.
- The user is only prompted to confirm a permission if the app being installed has a `<uses-permission>` element, the permission was already defined by some other app, and the protection level is not signature.

Custom access control

- On first access ask the user to grant access to API (see PendingIntents)
- Save package name, package signature to OpenKeychain (and optional settings for this specific app)
- Check against this database when clients bind to OpenKeychain's service:
  ```
  String[] callingPackages =
  getPackageManager().getPackagesForUid(
  Binder.getCallingUid());
  ```
- Let the user revoke access using a list of granted applications
- Trusted Intents implements similar access control mechanisms: `https://dev.guardianproject.info/projects/trustedintents`

UI

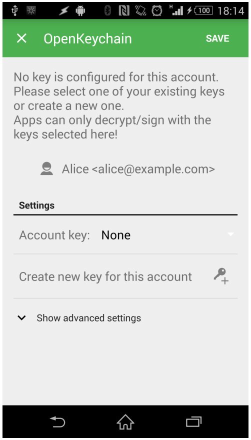

DISALLOW ACCESS
ALLOW ACCESS

The displayed app wants to encrypt/decrypt messages and sign them in your name.
Allow access?

WARNING: If you do not know why this screen appeared, disallow access! You can revoke access later using the 'Apps' screen.

OpenKeychain API Demo

Show advanced information


OpenKeychain
SAVE

No key is configured for this account. Please select one of your existing keys or create a new one.
Apps can only decrypt/sign with the keys selected here!

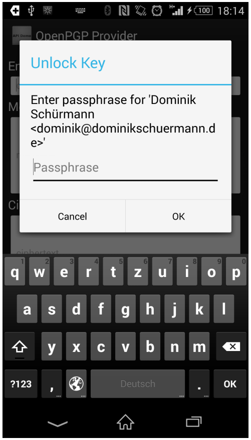
Alice <alice@example.com>

Settings

Account key: None

Create new key for this account

Show advanced settings

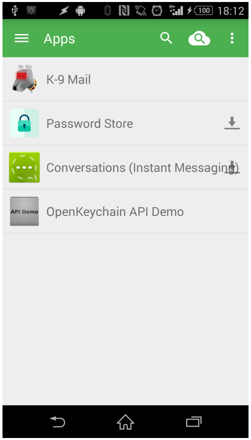

Unlock Key

Enter passphrase for 'Dominik Schürmann <dominik@dominikschuermann.de>'

Passphrase

Cancel
OK

UI


18:12

Apps

K-9 Mail

Password Store

Conversations (Instant Messaging)

OpenKeychain API Demo

Conclusion

- Sophisticated IPC APIs are possible
- Use generic AIDL method definitions
- Instead of method parameters, use Bundles with parameters
- For IPC objects use Parcelables
- Use versionable Parcelables

Conclusion

- Sophisticated IPC APIs are possible
- Use generic AIDL method definitions
- Instead of method parameters, use Bundles with parameters
- For IPC objects use Parcelables
- Use versionable Parcelables

Questions? Feedback?
Pull Requests for OpenKeychain?
`https://github.com/open-keychain/open-keychain`