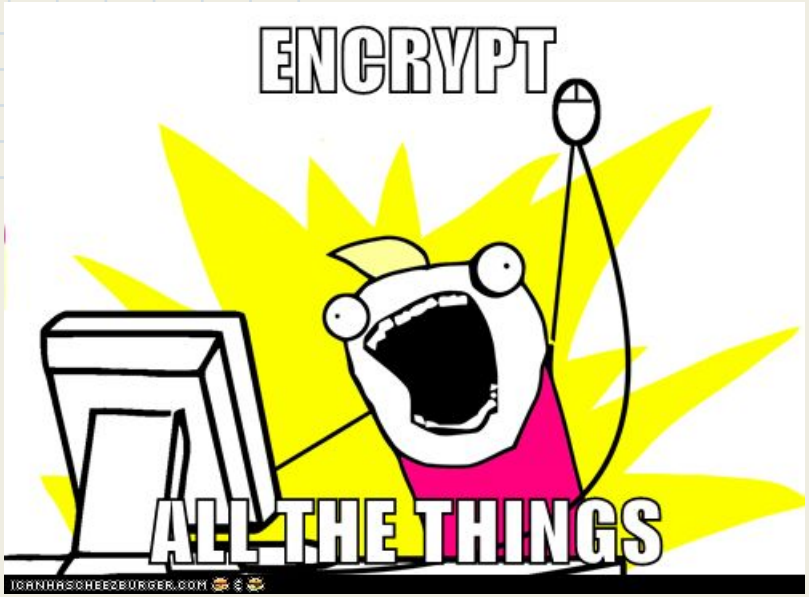
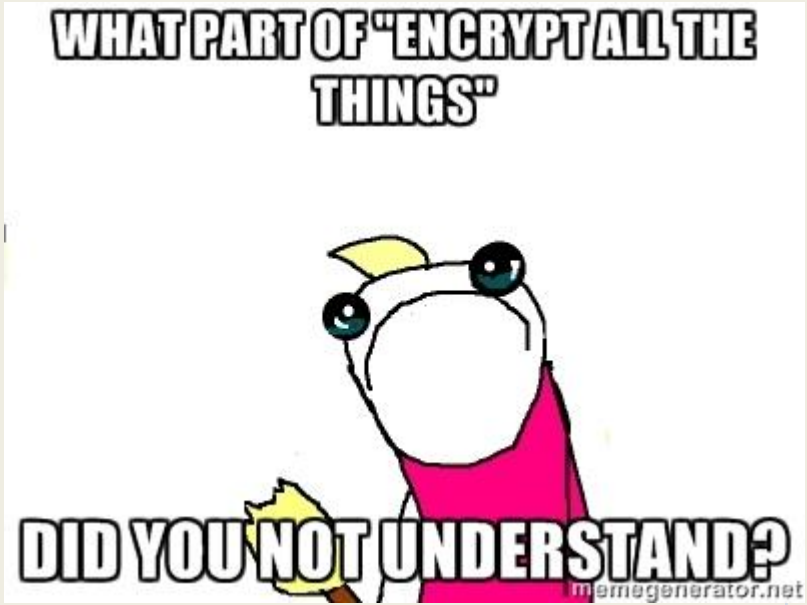
Encrypt All The Things:

Implementing App Mobile Security

Nathan Freitas
@n8fr8 @guardianproject
https://guardianproject.info





INTENTION **vs.** **EXECUTION**

The Guardian Project

https://guardianproject.info

Secure Your Mobile Life
Apps & Tools You Can Trust

The Guardian Project creates easy-to-use open source apps, mobile OS security enhancements, and customized mobile devices for people around the world to help them communicate more freely, and protect themselves from intrusion and monitoring.

Session Overview

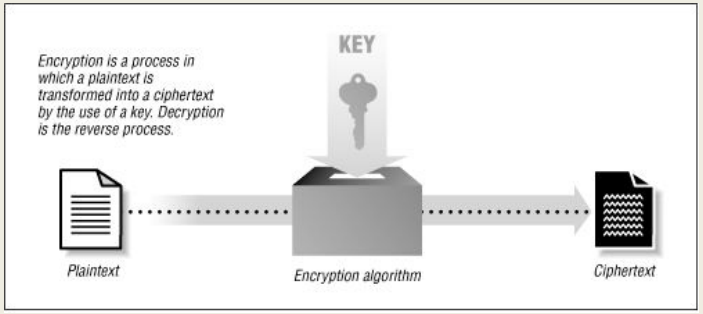
- **Overview** of Guardian Project Apps & Developer Libraries (30m)
- **Threat Models and War Stories:** Open Discussion about Risks, Fears and Security Needs (30m)
- **Encrypted Databases:** securing structured data in activities, services and content providers (1hr)
- **Encrypted Files:** securing arbitrary files from small to large (30m)
- **Secured Networking:** defending against man-in-the-middle, SSL stripping, filtering and more (30m)
- **Hands-On Implementation** time for sample work or debugging your own apps with new security features (1.5hr)

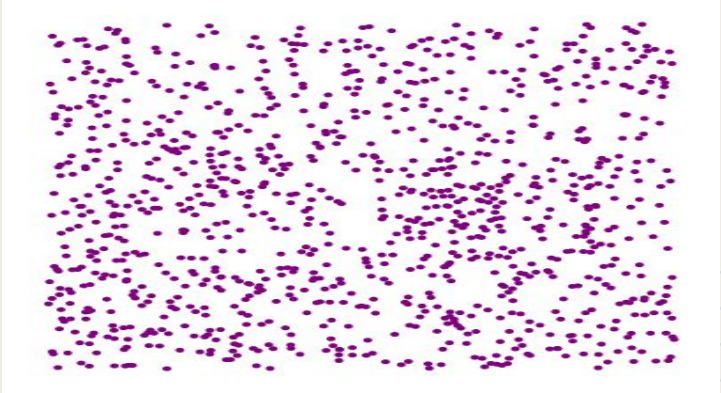
Encryption

a *very* quick introduction

What is Encryption?

- Plaintext + Algorithm + Key =Ciphertext
- Symmetric vs Asymmetric, Private vs Public
- Randomness:
  Actual vs Pseudo
- Common Cryptography Tools: OpenSSL, PGP (GnuPG!), BouncyCastle





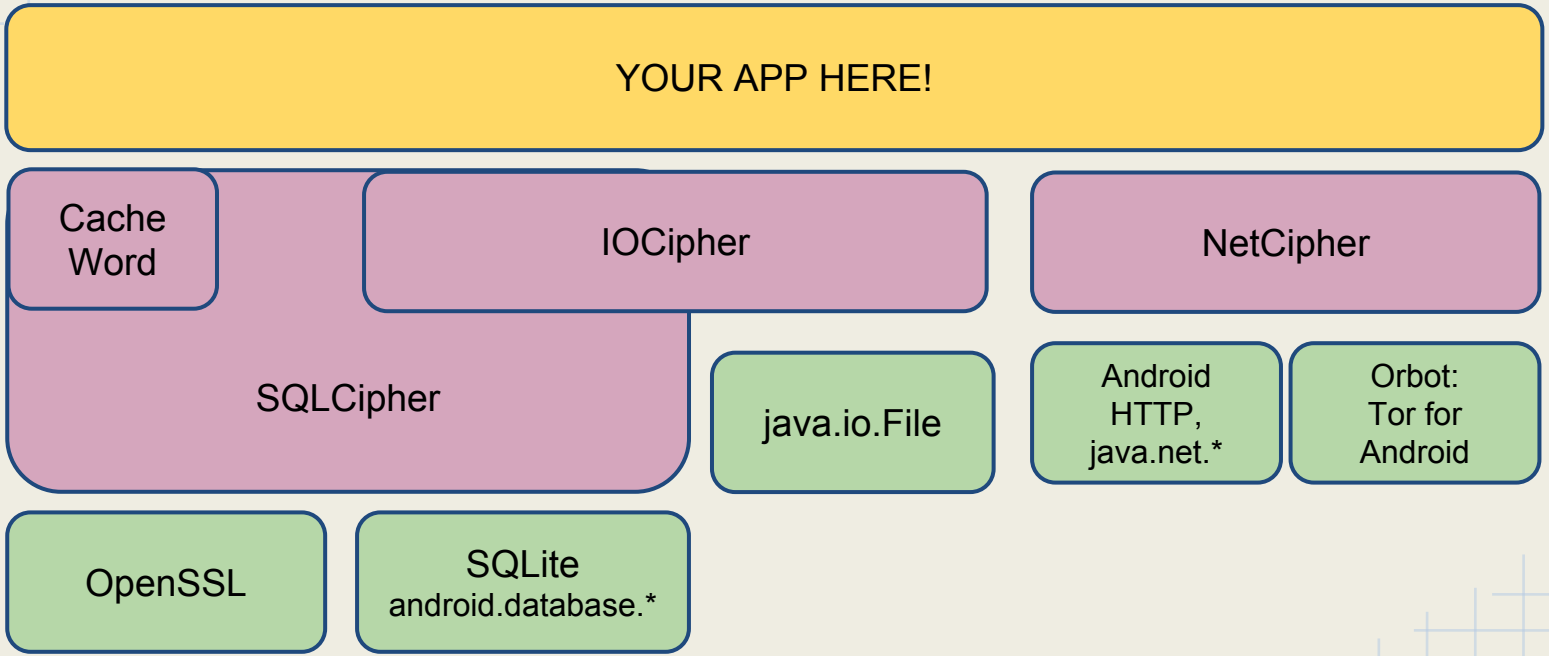
Android Built-in Encryption

- HTTPS / TLS / SSL
- javax.crypto “BouncyCastle”
- OpenSSL
- Full Disk Encryption
- Android KeyChain (> API 18)

CipherKit

https://guardianproject.info/code

CipherKit “Platform”



“CipherKit” Dev Libraries

CipherKit is designed for Android app developers to make apps that are able to ensure better privacy, security and anonymity

SQLCipher: Encrypted Database
SQLCipher is an SQLite extension that provides transparent 256-bit AES encryption of database files. It mirrors the standard android.database API. Pages are encrypted before being written to disk and are decrypted when read back.

IOCipher: Encrypted Virtual Disk
IOCipher is a virtual encrypted disk for apps without requiring the device to be rooted. It uses a clone of the standard java.io API for working with files. Just password handling & opening the virtual disk are what stand between developers and fully encrypted file storage. It is based on libsqlfs and SQLCipher.

NetCipher: Encrypted Network Data & Tor Integration
NetCipher is improving network security. It provides a strong TLS/SSL verifier to help mitigate weaknesses in the certificate authority system. It eases the implementation of supporting SOCKS and HTTP proxies into applications and also supports onion routing for anonymity and traffic surveillance circumvention.

Let’s take a step back...

(to figure out what it is we are worried about)

Basic Threat Modeling

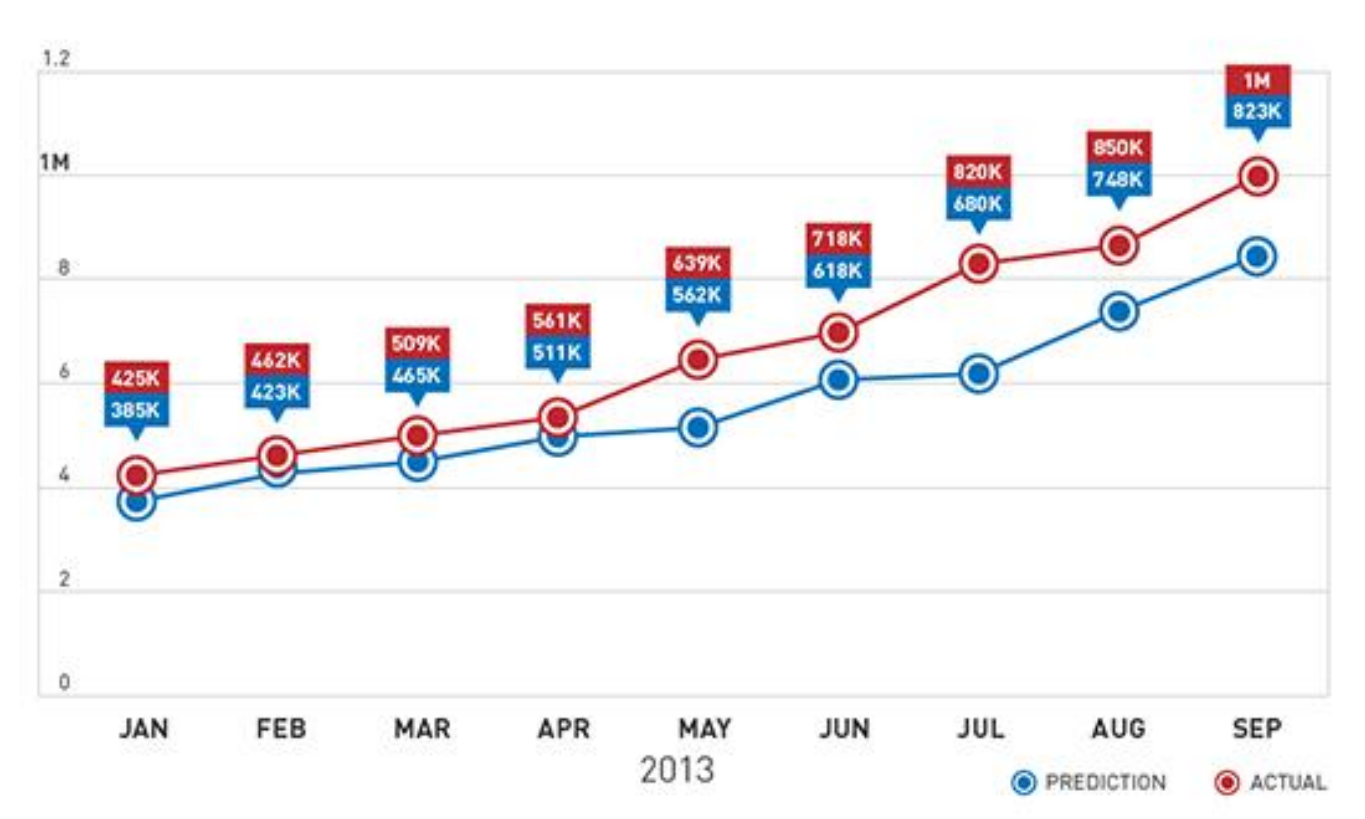
- “What are you worried about?”
  aka Possible Attack Vectors
- What data are you collecting or services are you providing that might be enticing or exposed?
- Are the potential threats you face coming from the device (other apps or physical access) or the network?

War Stories?

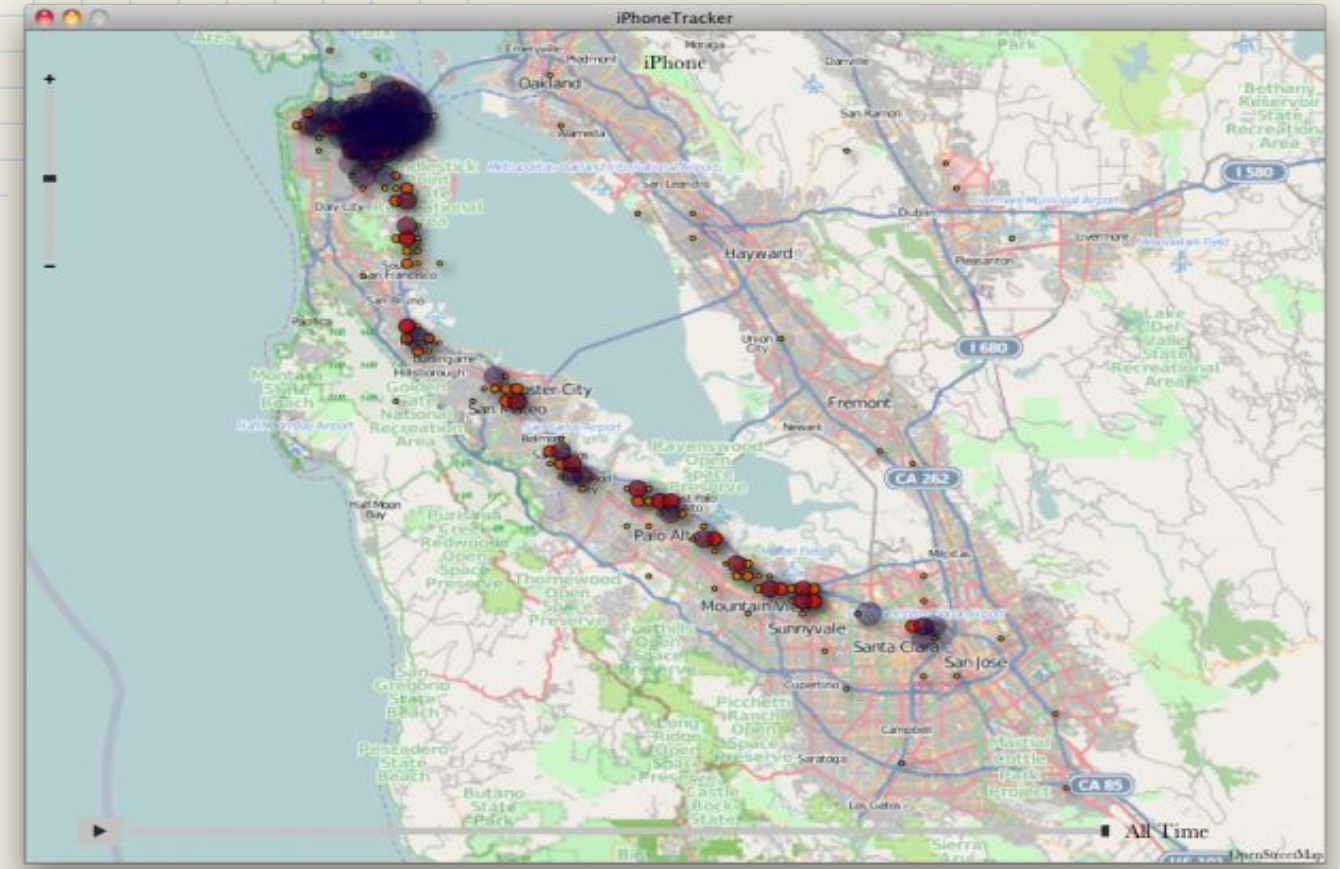
- Have your apps, your business or your users or customers lives or businesses been affected by malware or security breaches?
- Do you work in an industry that has specific requirements related to security and privacy?
- Do you target a region of the world where users might be more exposed to attack, surveillance or privacy violations?

Threat Landscape

- Forensic Analysis
- Rooting / Jail breaking
- OS Issues
- Infrequent Updates
- Removable Storage
- Cloud Services
- Targeted Attacks
- Device Sharing



Malware on the rise: http://blog.trendmicro.com/trendlabs-security-intelligence/mobile-malware-high-risk-apps-hit-1m-mark/



Cached GPS data stored in plain text
http://elifelog.org/book/iphone-gps-cache-data

UNIVERSAL FORENSIC EXTRACTION DEVICE

CELLEBRITE UFED FOR MOBILE FORENSICS

Addressing the growing need for fast and flexible mobile forensics, Cellebrite's Universal Forensic Extraction Device (UFED) is a stand-alone device for simple extraction of mobile phone data.

In use by military, law enforcement, governments, and intelligence agencies across the world, UFED enables recovery and analysis of invaluable evidentiary evidence from mobile phones.

Extracting data in a forensic manner and presenting it with the integrity of the data intact ensures that the evidence will be admissible in court.

AT A GLANCE

- Thorough extraction of mobile phone data, including contacts, SMS messages, photos, videos, call logs, audio files, ESN, IMEI, ICCID, IMSI and more
- Unrivalled coverage and compatibility, with support for more than 3000 phones
- Mapping of geotags on Google Maps
- SIM ID cloning circumvents missing SIM card and PIN-locked SIMs and neutralizes the phone from any network activity during analysis
- Field-ready mobile forensics – portable, fast and easy to operate, the Ruggedized UFED is battery powered and comes with all accessories needed for harsh field conditions

COMPREHENSIVE DATA EXTRACTION

Cellebrite ensures that almost all future devices are supported prior to retail launch. We work exclusively with more than 140 mobile operators – including T-Mobile, Verizon, Sprint, Orange, Vodafone, AT&T, Telstra, TIM and many others – to monitor handset adoption. We also work directly with phone manufacturers, receiving pre-production handset support prior to launch.

- Call logs, including SIM deleted call history
- Contacts
- Phone details (IMEI / ESN, phone number)
- ICCID and IMSI
- Text messages (SMS), including SIM deleted messages
- Photos
- Videos
- Audio files
- SIM location information: TMSI, MCC, MNC, LAC
- Image geotags

UFED REPORT MANAGER

The UFED Report Manager enables you to save, print, export, and analyze the extracted data.

Concise, easy-to-analyze reports are generated in HTML or XML format, providing an organized print-out for use as a courtroom reference.

Reports also include important fields such as time and date of extraction, agent or officer who performed the extraction and department and case number.

SIM ID CLONING

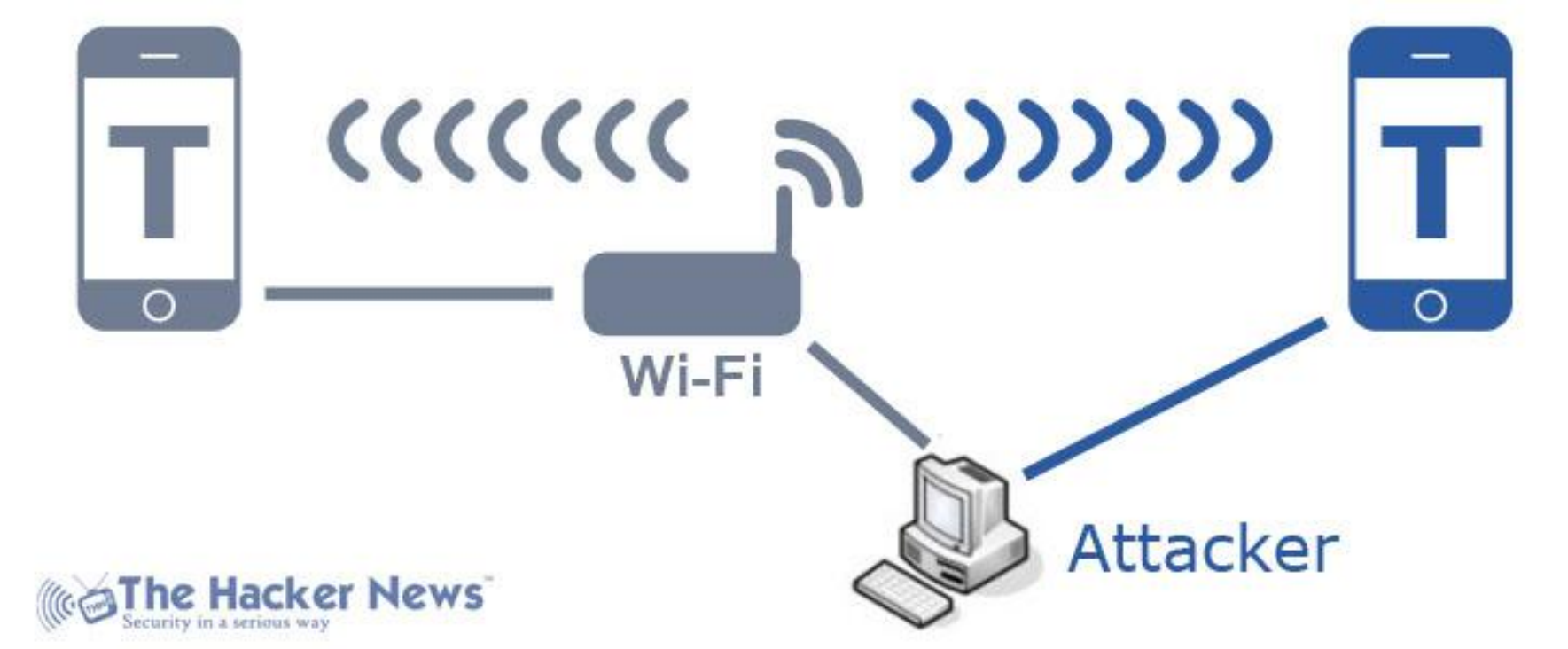
The UFED's SIM ID cloning feature allows data extraction from PIN locked SIMs, phones with missing SIM cards, and phones without network service.

The cloned SIM card also allows access phones without connecting to a network, preventing incoming calls and messages, while preserving the existing call and message history.

"Universal Forensic Extraction Devices" can quickly and easily copy all of the data from a mobile phone.

If tools like these fall into the wrong hands, it is easy to assume any unencrypted data on a device can be easily stolen.

Forensic Extraction
http://www.cellebrite.com/mobile-forensics



Man-in-the-Middle: http://thehackernews.com/2013/03/t-mobile-wi-fi-calling-app-vulnerable.html

Trust Levels

ID	Name	Description
1	Owner of the mobile phone	The primary operator of the mobile device. Assumed to have full access to the device, potentially secured with a PIN/password screen.
2	Detainer / criminal / bad actor	An authority figure or criminal who has or will be detaining the Owner[1]; has access to mobile phone. may have only manual/brute force access, or could have more sophisticated forensic extraction tools.
3	Operator of the mobile network	Access to call and message logs (sender/receiver/message content) and cell tower association data (rough location)
4	Employer, family or support organization;	May know the Owner[1]'s PIN/password, but otherwise has no access to data or network information; On the receiving end of an emergency message
5	Malicious App / Backdoor / Malware / Forensics App	Access to some or all of the the Owner[1]'s data depending upon app data permissions and encryption, as well as how full the backdoor is. Authorization is often required by the user to allow apps to access data.

Assets

ID	Name	Description	Trust Level
1	Personal data	Names, emails, phone numbers, calendar events, mostly stored on internal device memory	[1] Owner [5] Malicious App (as authorized)
2	Communication data	Text messages, emails, call logs, mostly stored on internal device memory	[1] Owner [3] Operator [5] Malicious App (as authorized)
3	Application data	Custom data stored by browsers, chat, social networking apps, on both internal and memory card;	[1] Owner [3] Operator (if not HTTP/S or SSL) [5] Malicious App (as authorized)
4	Media files	User generated and download photos, videos and music, primarily stored on memory card	[1] Owner [5] Malicious App

STRIDE Threat List

Type	Examples
Spoofing	- Detainer[2] or Malicious App[5] may gain control of mobile phone and pretend to be Owner[1]
Tampering	- Malicious App[5] changes configuration data on the device
Repudiation	- Malicious App[5] or other system backdoor may disable or block app - Operator[3] may passively monitor messages and pass the information along to the Detainer[2]
Information Disclosure	Detainer[2] could have full access to Assets stored on the mobile device - Detainer[2] may have physical and logical forensic data extraction tools that can override password controls on device and read from "wiped" storage - Operator[3] may learn identity of Support Org[4]
Denial of Service	- Communications may be blocked from being sent or received by Operator [3] - Mobile phone may be disabled by Operator[3] or Malicious App[5] from running remote wipe
Elevation of Privilege	- Malicious App [5] launches insecured intents or exploits known bug - Detainer[2] or Operator[3] may be able to impersonate the Owner[1]

Security Controls / Mitigation

Type	Tactics
Authentication (vs. Spoofing)	- Create a a non obvious passphrase for use in app - Lock screen of your mobile phone using passphrase or PIN
Authorization & Auditing (vs Tampering, Repudiation, Elevation of Priv)	- Do not install any unnecessary, third-party mobile apps with network access - Scan your mobile device using available malware tools - Install a firewall or network connection monitoring utility - Use a non-real name registered SIM card and mobile phone
Cryptography and Identity Protection (vs Information Disclosure)	- For extra sensitive data, use an app that supports an and password authentication and encrypted database - Use a mobile OS with disk and memory card encryption - Use only browser-based HTTPS services that do not store data locally - Do not store or save web service passwords on your mobile phone
Alternate Communications (vs Denial of Service)	- Use VPNs or Tor proxying software to hide source IP and traffic - Use apps/services that work in WIFI only mode if data service disabled - Use apps that allow device-to-device data sharing

SQLCipher

Encrypted Database

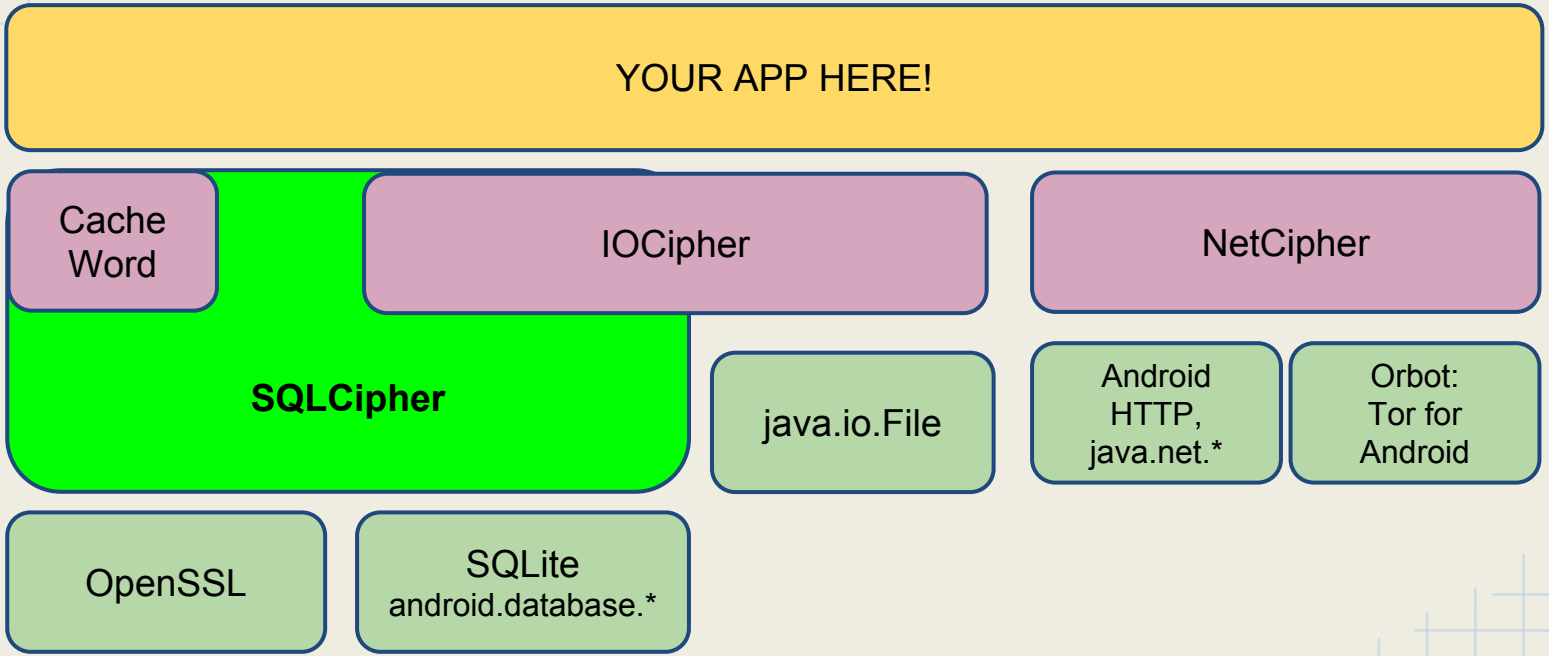
SQLCipher: Encrypted DB

SQLCipher is an SQLite extension that provides transparent 256-bit AES encryption of database files. It mirrors the standard android.database API. Pages are encrypted before being written to disk and are decrypted when read back.

SQLCipher has a small footprint and great performance so it's ideal for protecting embedded application databases and is well suited for mobile development.

- Blazing fast performance with as little as 5-15% overhead for encryption
- 100% of data in the database file is encrypted
- Uses good security practices (CBC mode, key derivation)
- Zero-configuration and application level cryptography
- Algorithms provided by the peer reviewed OpenSSL crypto library.

CipherKit “Platform”



Defense in Depth

Make attacks difficult with multiple layers of security

Principle of Least Privilege

Access to device should not allow access to all apps and data

Data Security

Minimize impact of unauthorized access, on and off device

Strategies

1. Authentication
2. Encryption
3. Authenticity

```
~ sjlombardo$ hexdump -C sqlite.db
00000000 53 51 4c 69 74 65 20 66 6f 72 6d 61 74 20 33 00 |SQLite format 3.|
…
000003c0 65 74 32 74 32 03 43 52 45 41 54 45 20 54 41 42 |et2t2.CREATE TAB|
000003d0 4c 45 20 74 32 28 61 2c 62 29 24 01 06 17 11 11 |LE t2(a,b)$…..|
…
000007e0 20 74 68 65 20 73 68 6f 77 15 01 03 01 2f 01 6f | the show…./.o|
000007f0 6e 65 20 66 6f 72 20 74 68 65 20 6d 6f 6e 65 79 |ne for the money|

~ $ sqlite3 sqlcipher.db
sqlite> PRAGMA KEY='test123';
sqlite> CREATE TABLE t1(a,b);
sqlite> INSERT INTO t1(a,b) VALUES ('one for the money', 'two for the show');
sqlite> .quit

~ $ hexdump -C sqlite.db
00000000 84 d1 36 18 eb b5 82 90 c4 70 0d ee 43 cb 61 87 |.?6.?..?p.?C?a.|
00000010 91 42 3c cd 55 24 ab c6 c4 1d c6 67 b4 e3 96 bb |.B?..?|
00000bf0 8e 99 ee 28 23 43 ab a4 97 cd 63 42 8a 8e 7c c6 |..?(#C??.?cB..|?|

~ $ sqlite3 sqlcipher.db
sqlite> SELECT * FROM t1;
Error: file is encrypted or is not a database
```

```
import net.sqlcipher.database.SQLiteDatabase;

SQLiteDatabase.loadLibs(this);

SQLiteDatabase db = eventsData.getWritableDatabase
(“mypassword”);
```

https://github.com/sqlcipher/android-database-sqlcipher

Simple Steps

We've packaged up a very simple SDK for any Android developer to add SQLCipher into their app with the following three steps:

1. Add a single sqlcipher.jar and a few .so's to the application libs directory
2. Update the import path from *android.database.sqlite.** to*info.guardianproject.database.sqlite.** in any source files that reference it. The original android.database.Cursor can still be used unchanged.
3. Init the database in onCreate() and pass a variable argument to the open database method with a password*:

- SQLiteDatabase.loadLibs(this); //first init the db libraries with the context
- SQLiteOpenHelper.getWritableDatabase("thisismysecret"):

Features

- AES 256 CBC
- Random IVs
- Random salt
- Key Derivation
- MAC
- OpenSSL
- Fast startup
- No size limit

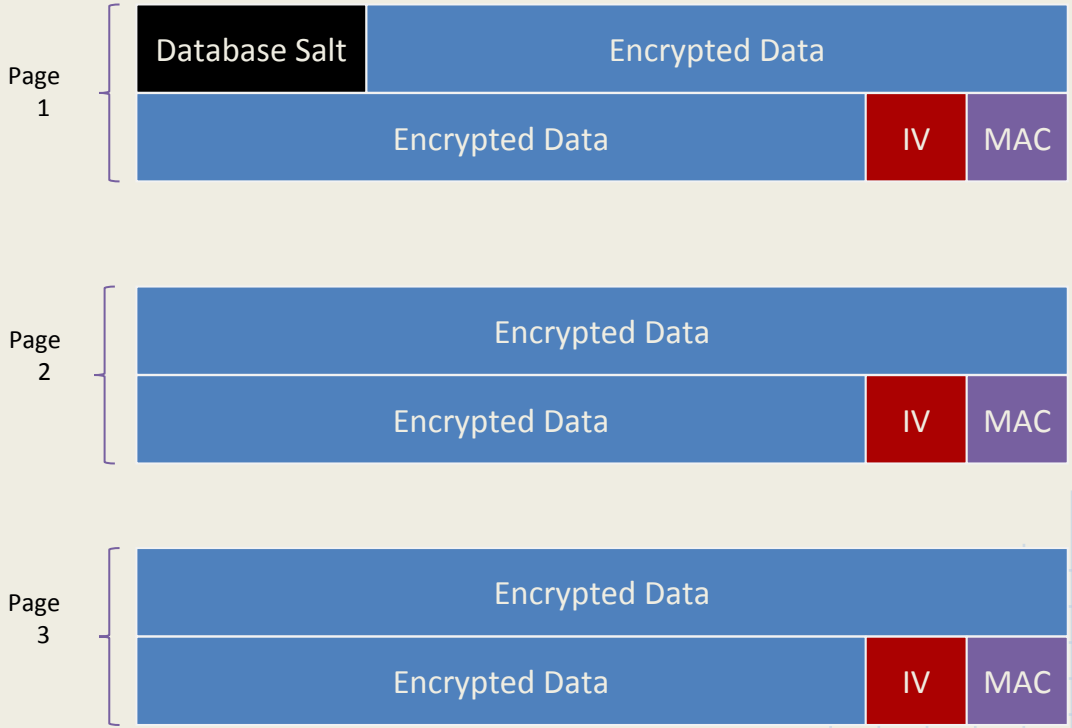
How it Works

Pager Codec

Key Derivation

Encryption

MAC



Performance

Create Table (1st operation)

Normal (ms)	Encrypted (ms)	Difference
61	142	132.8%

```
CREATE TABLE t1(a INTEGER, b INTEGER, c VARCHAR(100));
```

500 Inserts (no transaction)

Normal (ms)	Encrypted (ms)	Difference
20832	24414	17.2%

```
INSERT INTO t1 VALUES (@a,@b,@c);
```

30000 Inserts (with transaction)

Normal (ms)	Encrypted (ms)	Difference
11002	11281	2.5%

```
INSERT INTO t2 VALUES (@a,@b,@c);
```

500 Updates (w/o index, w/o transaction)

Normal (ms)	Encrypted (ms)	Difference
37986	39164	3.1%

```
UPDATE t2 SET b=b*2 WHERE a = @a
```

30000 Selects (w/ index)

Normal (ms)	Encrypted (ms)	Difference
5334	5498	3.1%

```
SELECT * FROM t2 WHERE a = @a
```

2500 Updates (w/ index + transaction)

Normal (ms)	Encrypted (ms)	Difference
1214	1373	13.1%

```
UPDATE t2 SET b = @b WHERE a = @a
```

- PRAGMA rekey
- PRAGMA cipher
- PRAGMA kdf_iter
- PRAGMA cipher_page_size
- PRAGMA cipher_use_hmac
- ATTACH
- sqlcipher_export()

Advanced

IOCipher

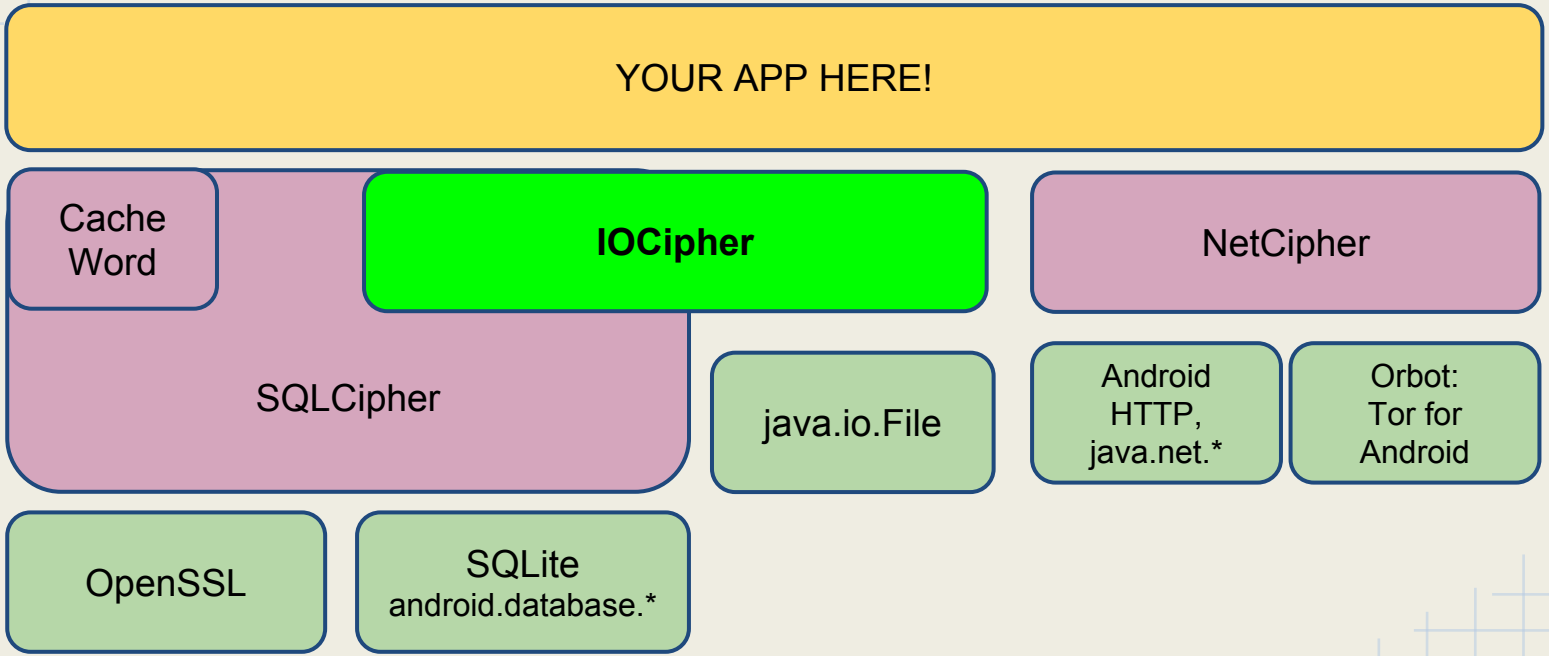
Encrypted Virtual File System

IOCipher: Encrypted Files

IOCipher provides a virtual encrypted disk for Android apps without requiring the device to be rooted. It uses a clone of the standard java.io API for working with files, so developers already know how to use it. Only password handling, and opening the virtual disk are what stand between the developer and working encrypted file storage. It is based on and SQLCipher.

IOCipher is a cousin to SQLCipher-for-Android since it is also based on SQLCipher and uses the same approach of repurposing an API that developers already know well. It is built on top of libsqlfs, a filesystem implemented in SQL that exposes a FUSE API.

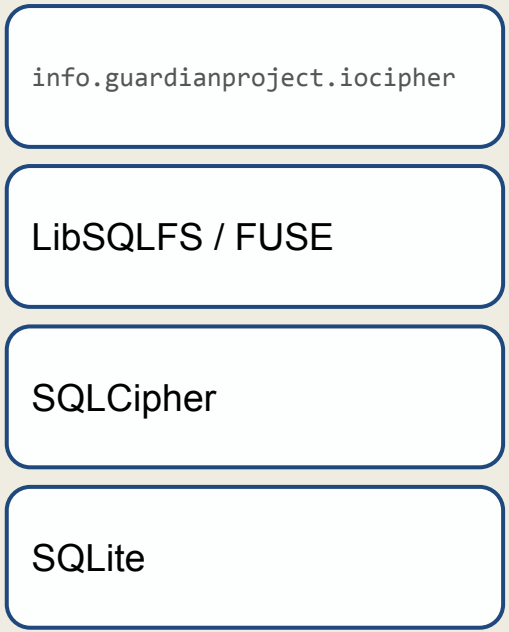
CipherKit “Platform”



IOCipher: Core Features

- Secure transparent app-level virtual encrypted disk
- No root required
- Only three new methods to learn: new VirtualFileSystem(dbFile), VirtualFileSystem.mount(password), and VirtualFileSystem.unmount()
- Supports Android versions 2.1 and above
- Licensed under the LGPL v3+

IOCipher: The Stack



Java/JNI wrapper API

Virtual Filesystem that maps to SQL schema / structured database

Encryption layer for SQLite

Base storage mechanism

Adding IOCipher to App

- manage the password
- connect to your encrypted disk's file using new VirtualFileSystem(dbFile)
- mount it with a password using VirtualFileSystem.mount(password)
- replace the relevant java.io import statements withinfo.guardianproject.iocipher, e.g.:
  - import info.guardianproject.iocipher.File;
  - import info.guardianproject.iocipher.FileOutputStream;
  - import info.guardianproject.iocipher.FileReader;
  - import info.guardianproject.iocipher.IOCipherFileChannel;
  - import info.guardianproject.iocipher.VirtualFileSystem;
  - import java.io.FileNotFoundException;
  - import java.io.IOException;
  - import java.io.InputStream;
  - import java.nio.channels.Channels;
  - import java.nio.channels.ReadableByteChannel;

```
import info.guardianproject.iocipher.File;
import info.guardianproject.iocipher.FileOutputStream;
import info.guardianproject.iocipher.VirtualFileSystem;

File dbFile = getDir("vfs", MODE_PRIVATE).getAbsolutePath() + "/myfiles.db";

vfs = new VirtualFileSystem(dbFile);

// TODO don't use a hard-coded password! prompt for the password
vfs.mount("my fake password");

File file = new File(dirPath);
File[] files = file.listFiles();
```

https://github.com/guardianproject/IOCipherExample

CacheWord

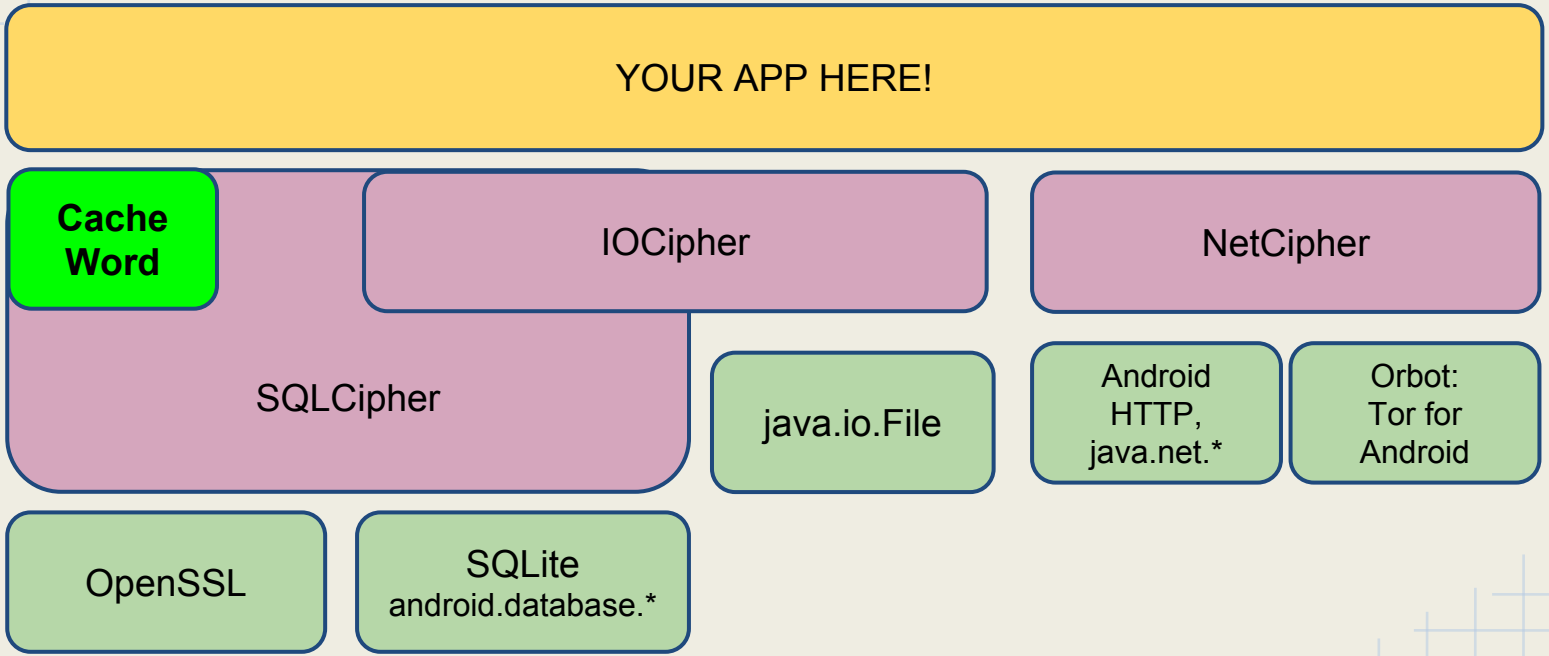
Secure Passphrase Management

CacheWord

CacheWord is an Android library project for passphrase caching and management. It helps app developers securely generate, store, and access secrets derived from a user's passphrase.

1. Secrets Management: how the secret key material for your app is generated, stored, and accessed
2. Passphrase Caching: store the passphrase in memory to avoid constantly prompting the user

CipherKit “Platform”

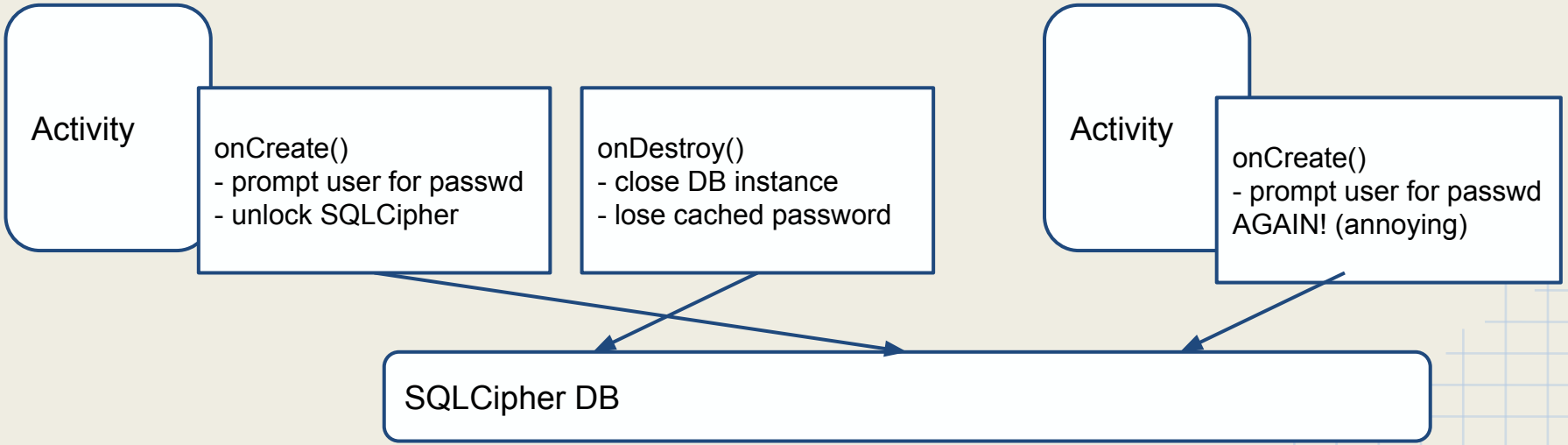


CacheWord Features

CacheWord manages key derivation, verification, persistence, passphrase resetting, and caching secret key material in memory.

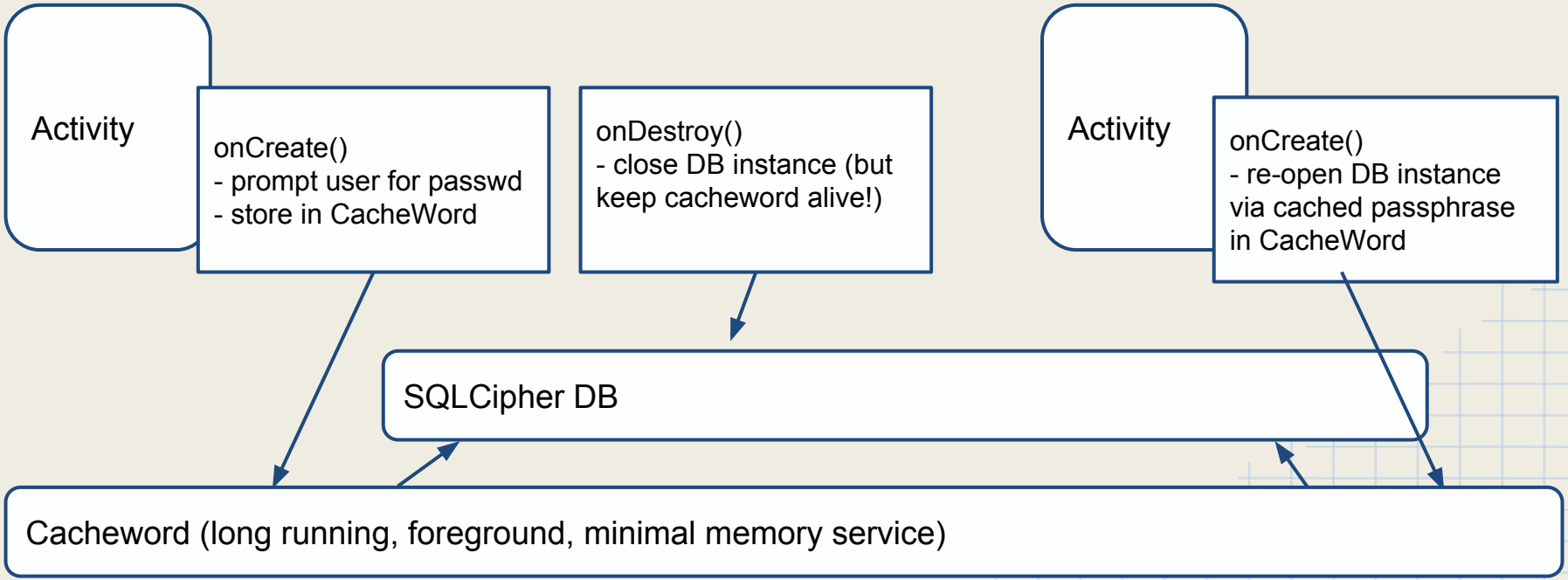
- Strong key derivation (PBKDF2)
- Secure secret storage (AES-256 GCM)
- Persistent notification: informs the user the app data is unlocked
- Configurable timeout: after a specified time of inactivity the app locks itself
- Manual clearing: the user can forcibly lock the application
- Uses Android's Keystore on 4.x if available - *Not Yet Implemented*

The Problem with Android...



(Activity, Service and even App lifespan is unpredictable)

Cacheword Solution



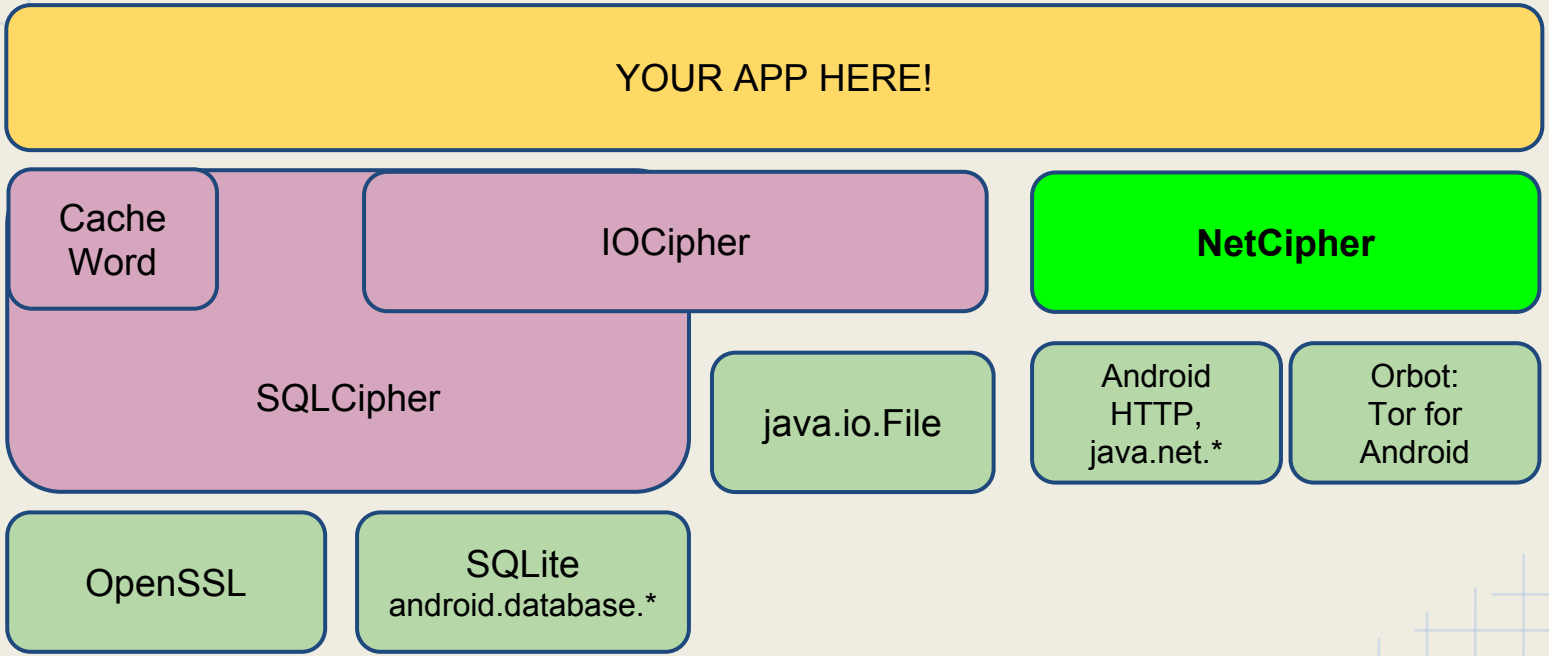
```
public class CacheWordSampleActivity extends Activity implements      ICacheWordSubscriber {
…
        mCacheWord = new CacheWordActivityHandler(this);

@Override
    public void onCacheWordLocked() {}

    @Override
    public void onCacheWordOpened() {
            // fetch the encryption key from CacheWordService
        SecretKey key = ((PassphraseSecrets) mCacheWord.getCachedSecrets()).getSecretKey();
    }

    @Override
       public void onCacheWordUninitialized() {
               mCacheWord.setCachedSecrets(PassphraseSecrets.initializeSecrets(
                       CacheWordSampleActivity.this, “my secret passphrase”));
           }
```

https://github.com/guardianproject/cacheword/tree/master/sample

NetCipher

Secured Networking

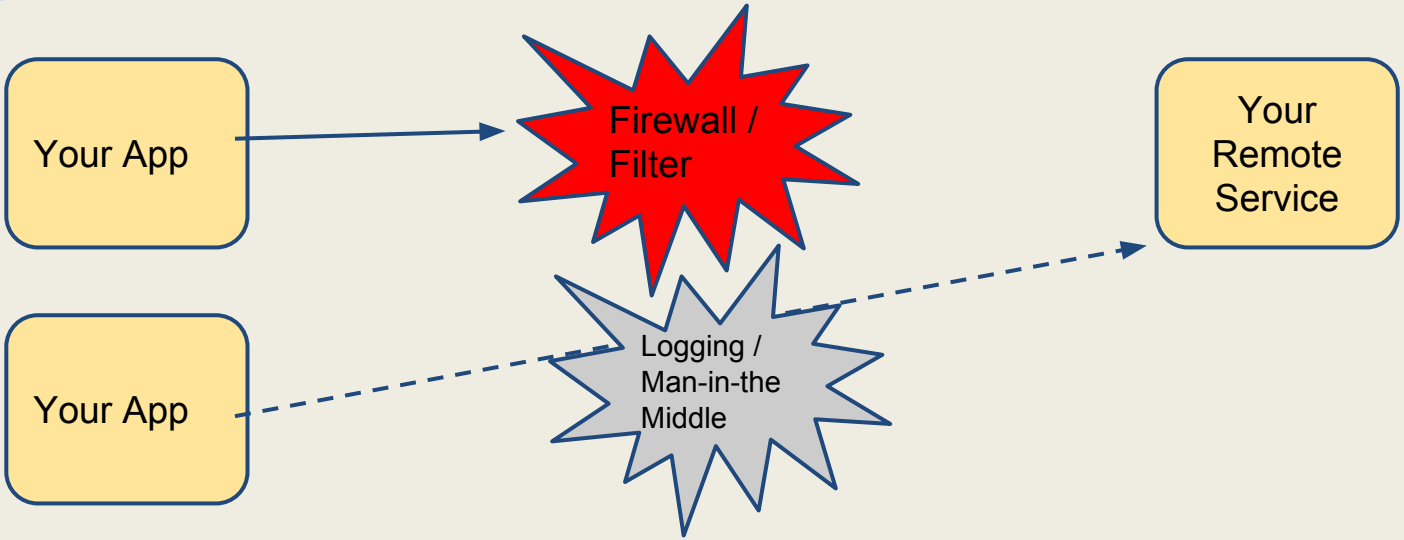
CipherKit “Platform”



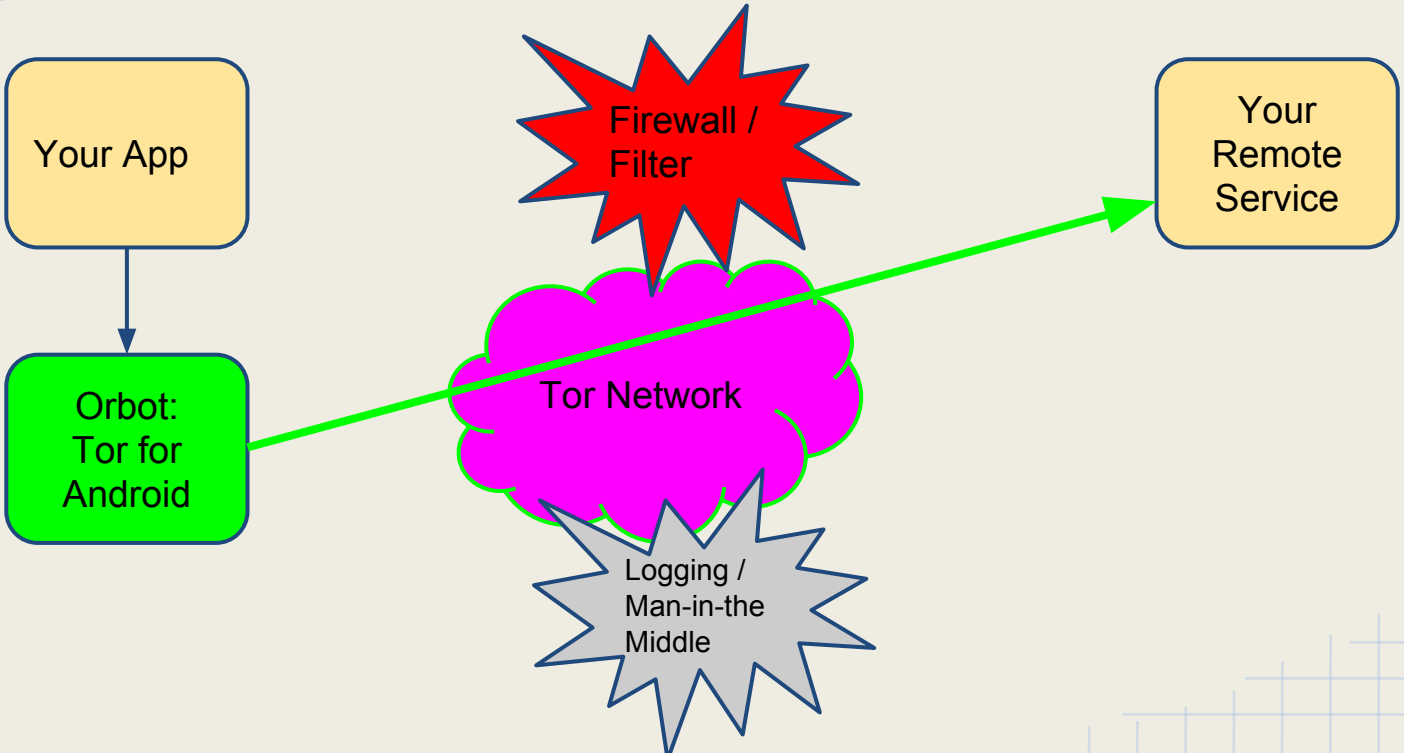
NetCipher: 3 reasons

1. **Stronger Sockets**: Through support for the right cipher suites, pinning and more, we ensure your encrypted connections are as strong as possible.
2. **Proxied Connection Support**: HTTP and SOCKS proxy connection support for HTTP and HTTP/S traffic through specific configuration of the Apache HTTPClient library
3. **OrbotHelper**: a utility class to support application integration with Orbot: Tor for Android. Check if its installed, running, etc.

Network Threats



NetCipher: Tor Proxying



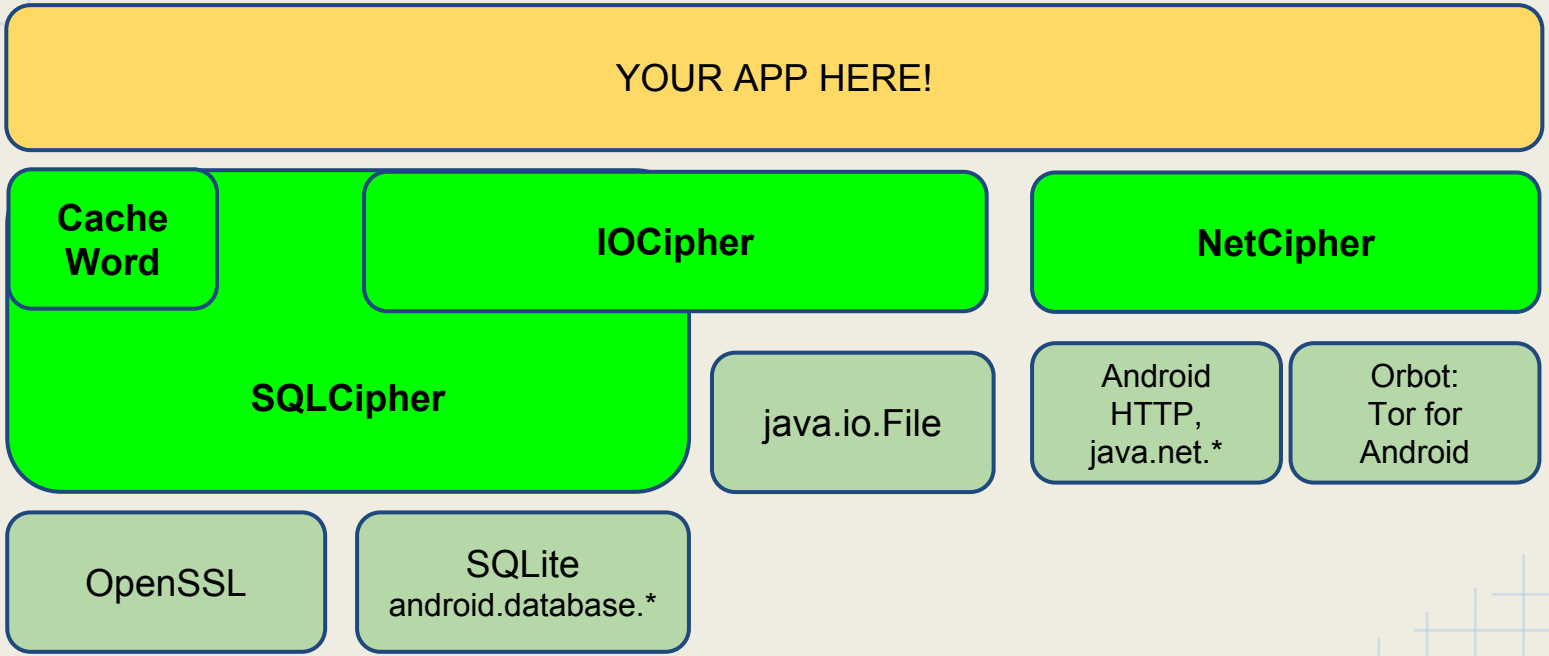
```
OrbotHelper oc = new OrbotHelper(this);
      if (!oc.isOrbotInstalled())
          oc.promptToInstall(this);
      else if (!oc.isOrbotRunning())
           oc.requestOrbotStart(this);

StrongHttpsClient httpclient = new StrongHttpsClient(getApplicationContext());

      if (pType == null)
          httpclient.useProxy(false, null, null, -1);
      else if (pType == Proxy.Type.SOCKS)
          httpclient.useProxy(true, "SOCKS", proxyHost, proxyPort);
      else if (pType == Proxy.Type.HTTP)
          httpclient.useProxy(true, ConnRoutePNames.DEFAULT_PROXY, proxyHost, proxyPort);
```

https://github.com/guardianproject/NetCipher

Hands-On Time!

Work with Samples or Your Own App

Time to encrypt all the things!



Questions?

What haven’t we covered?

From here...

https://guardianproject.info/contact

Guardian-Dev and SQLCipher mailing lists
IRC (freenode): #guardianproject
Project Trackers: https://dev.guardianproject.info

support@guardianproject.info