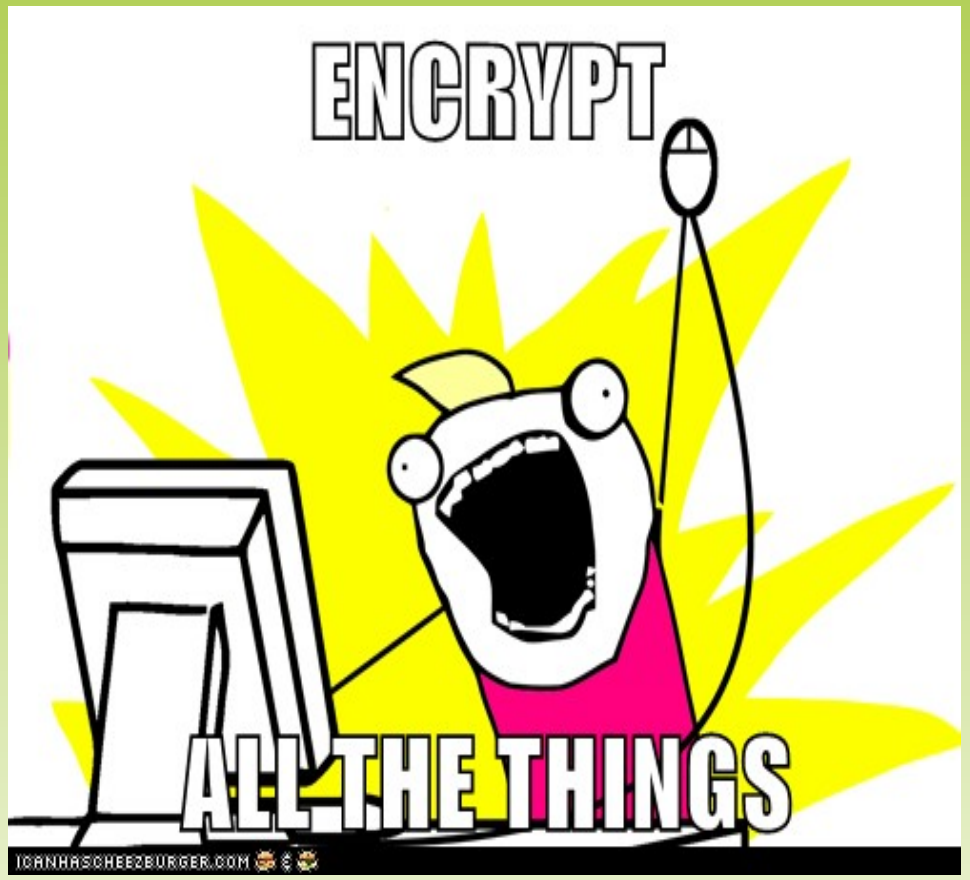
droidcon NYC 2014

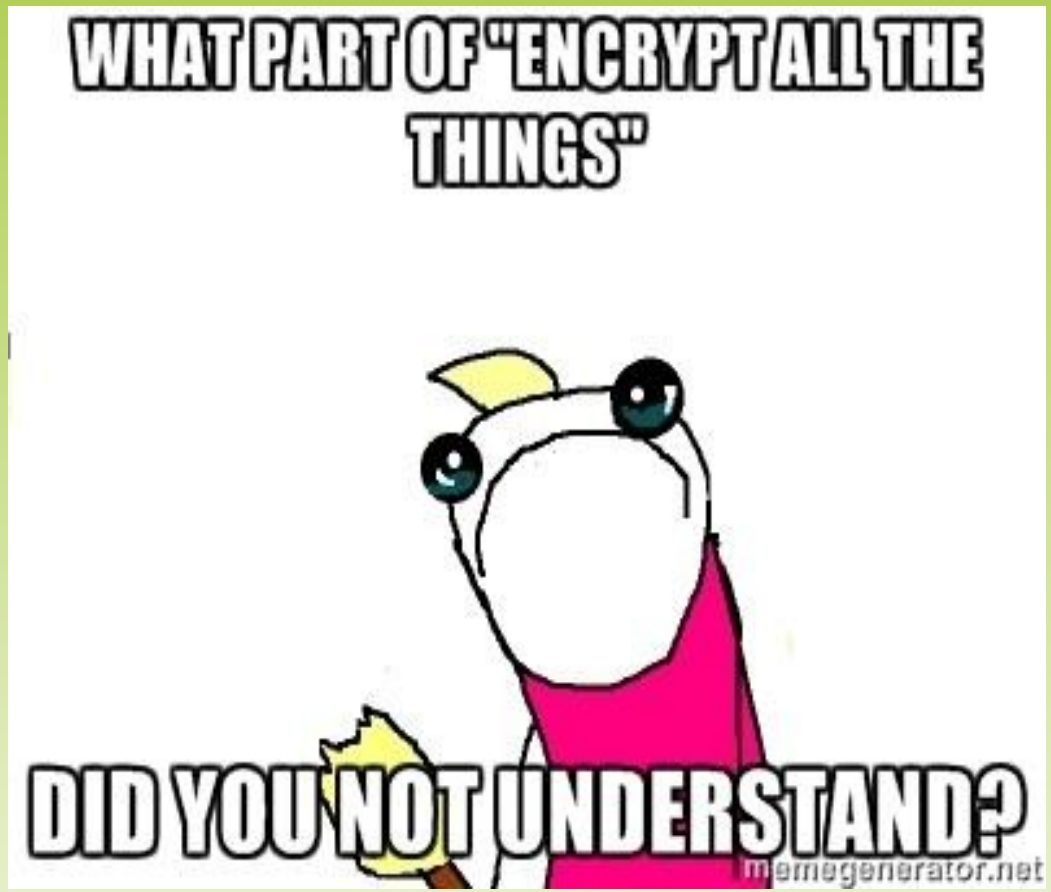
Encrypt All The Things!

xkcd comics reproduced under CC license from Randall Munroe. Beware of Hat Guy.

Your Users' Bad Guys

- Different Users Have Different Security Concerns
- Do Not Assume All Users Are the Same as You
  - …unless you are your only user
- Your Users' Collective Threats = Your Threats





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

Rest and Motion

Followed By More Rest, Because Motion is Tiring

- Securing Data at Rest = Local Storage
  - Databases
  - SharedPreferences
  - Other Types of Files
- Securing Data in Motion = Internet (mostly)
  - SSL
  - OTR

The Droid Is Not Enough

- Lock Screen?
  - Mechanical brute forcing
- Internal Storage?
  - Rooting
- Full-Disk Crypto?
  - Digital brute forcing

Your Objectives (One Hopes)

- Cheap and Easy Security
  - Only have so much time to budget
  - Aiming for "low hanging fruit"
- Effective Security
  - "Using CryptoLint, we performed a study on cryptographic implementations in 11,748 Android applications. Overall we find that 10,327 programs – 88% in total – use cryptography inappropriately. The raw scale of misuse indicates a widespread misunderstanding of how to properly use cryptography in Android development."

You're Doing It Wrong

- Hardcoded Passphrases
- Manually Seeding SecureRandom
  - …with a hardcoded seed
- Hardcoded Salts
- Insufficient Key Generation Iterations
- Non-Random Initialization Vectors

Introducing SQLCipher

- SQLCipher
  - Modified version of SQLite
  - AES-256 encryption by default, of all data
  - Relatively low overhead
  - Cross-platform
  - BSD license

Introducing SQLCipher

- SQLCipher Security
  - Customizable encryption algorithm
    - Based on OpenSSL libcrypto
  - Individual pages encrypted, with own initialization vector
  - Message authentication code (MAC) per page, to detect tampering
  - Hashed passphrase (PBKDF2) for key
    - 4,000 iterations, moving to 64,000 for 3.0

Introducing SQLCipher

- SQLCipher for Android
  - NDK-compiled binaries
  - Drop-in replacement classes for Android's SQLite classes
    - SQLiteDatabase
    - SQLiteOpenHelper
    - Etc.
  - Modify your code, third-party libraries also using SQLite

Integrating SQLCipher

- Step #1: Add to Project
  - Download ZIP file from: `http://sqlcipher.net/downloads/`
  - Copy ZIP's `assets/` into project's `assets/`
  - Copy ZIP's `libs/` into project's `libs/`

Integrating SQLCipher

- Step #2: Replace Import Statements
  - Eclipse
    - Delete all `android.database.*` and `android.database.sqlite.*` imports
    - Use Ctrl-Shift-O and choose the `net.sqlcipher` equivalents

Integrating SQLCipher

- Step #2: Replace Import Statements
  - Outside of Eclipse
    - Replace all occurrences of `android.database` with `net.sqlcipher`, revert back as needed
    - Replace all occurrences of `android.database.sqlite` with `net.sqlcipher.database`

Integrating SQLCipher

- Step #3: Supply Passphrases
  - SQLiteDatabase `openOrCreateDatabase()`, etc.
  - SQLiteOpenHelper `getReadableDatabase()` and `getWritableDatabase()`
  - Collect passphrase from user via your own UI

Integrating SQLCipher

- Step #4: Testing
  - Tests should work when starting with a clean install
    - No existing unencrypted database
- Step #5: Beer!
  - Hooray, beer!

Integrating SQLCipher

- Other Integration Issues
  - Upgrading to encryption
  - ContentProvider
    - Can work, but need to get passphrase to it before using the database (e.g., `call()`)

Integrating SQLCipher

- About the Bloat
  - 4MB base
  - Additional ~5MB for x86
  - Additional ~3MB for ARM
  - Why?
    - Complete independent copy of SQLite
    - Static library implementation of OpenSSL
    - Independent copy of ICU collation ruleset

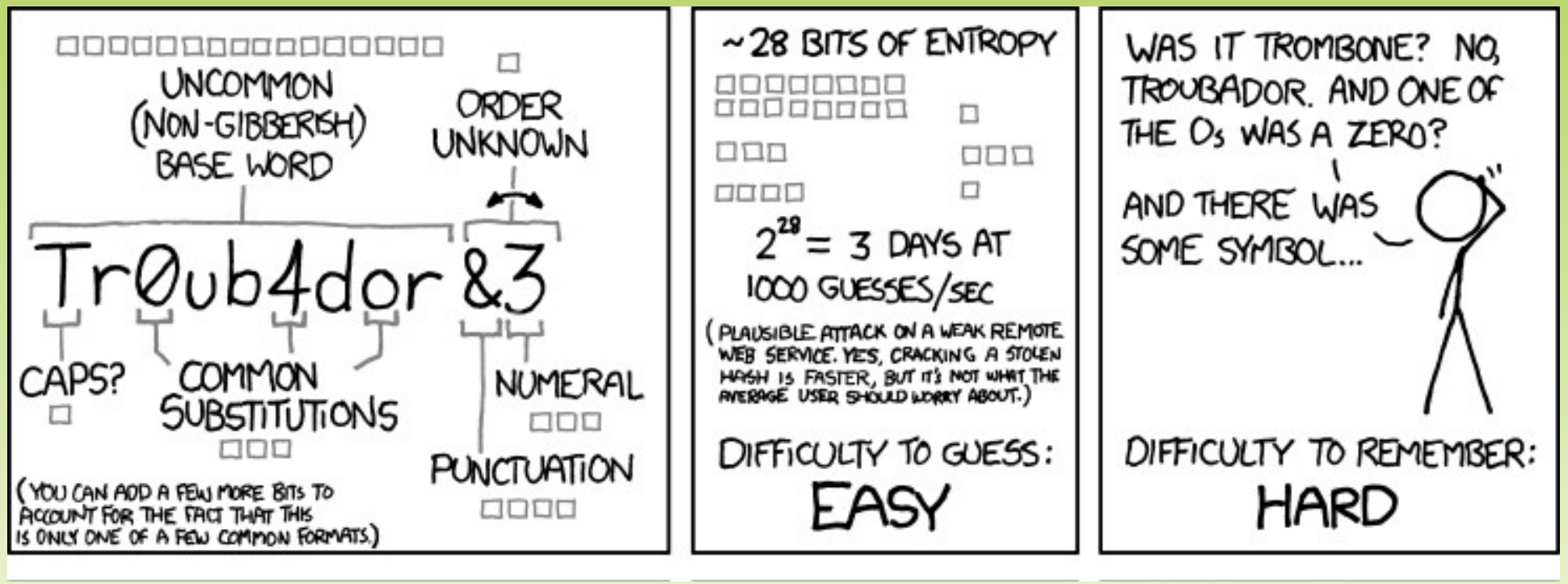
Passphrases

- Use Case: Defending the User's Data
  - User knows and supplies passphrase
- Non-Use Case: DRM
  - Developer knows and supplies passphrase
  - Trivially hackable to discover a baked-in passphrase
  - Rooted device users can get at generated passphrases

Passphrases

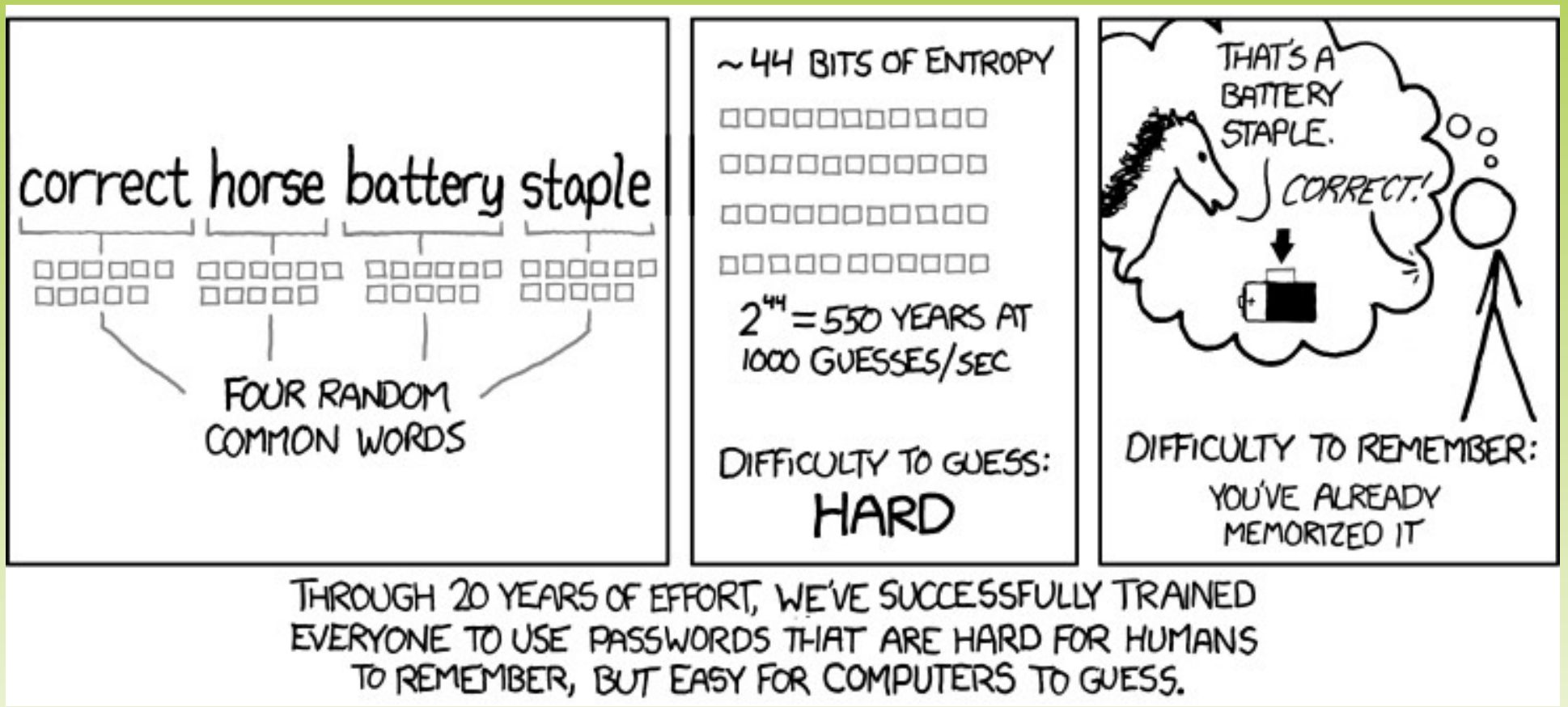
- Passphrase Entry Pain
  - Users do not like typing long passwords
  - Result = weaker quality
  - Option: "diceware"
    - Choose ~6 words from stock list
    - Can offer scrolling lists, auto-complete to help speed data entry
    - Downside: more annoying for accessibility

Passphrases



xkcd comics reproduced under CC license from Randall Munroe, even though Hat Guy owns a $5 wrench

Passphrases



xkcd comics reproduced under CC license from Randall Munroe, but BYO talking horse

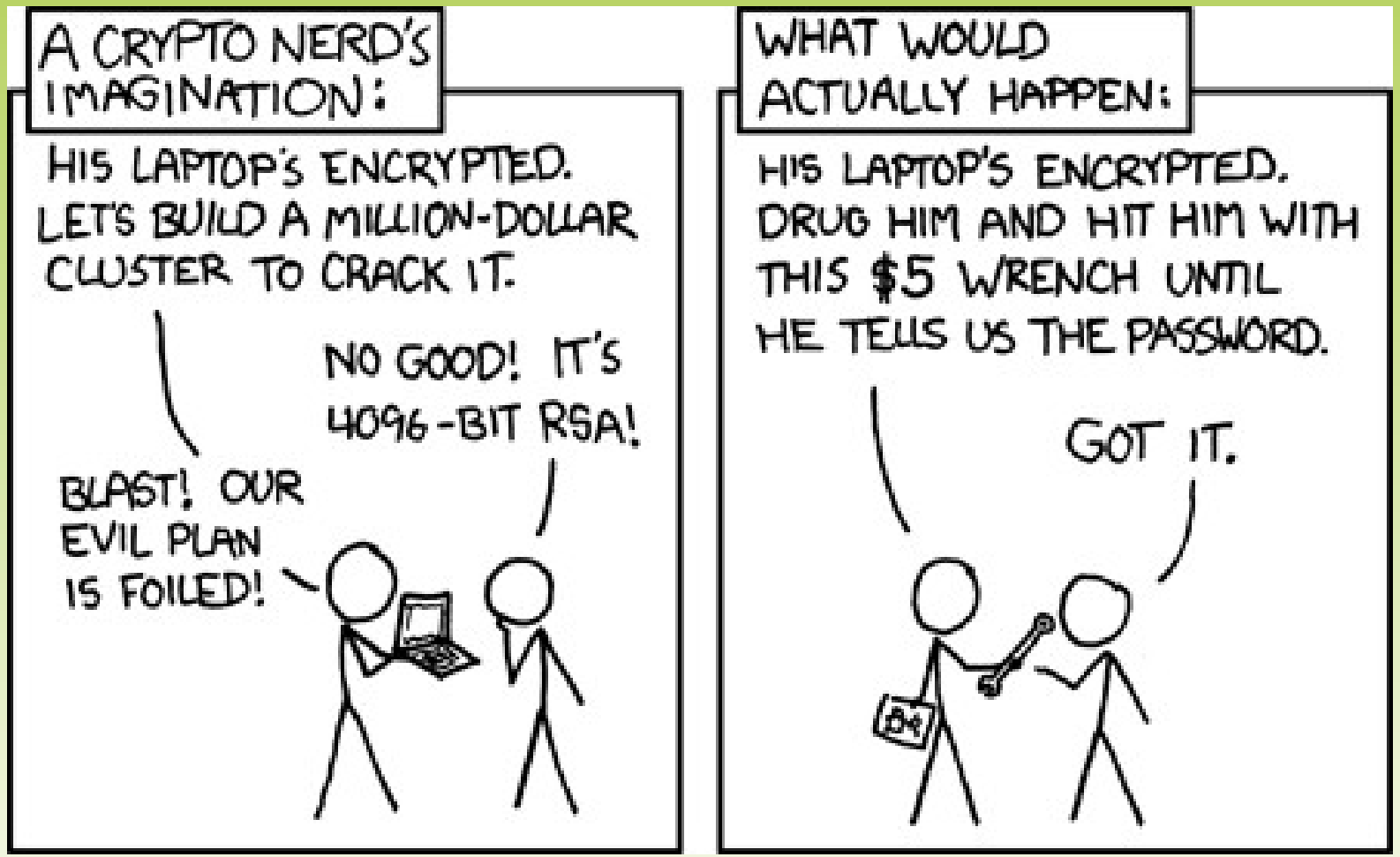
Passphrases

- Multi-Factor Authentication
  - Passphrase generated in code from user-supplied pieces
  - Organization options
    - Simple concatenation
    - Concatenation with factor prefix, un-typeable divider characters

Passphrases

- Multi-Factor Authentication Objectives
  - Longer passphrase without as much user input
  - Help defeat casual attacks
    - Need all factors to access via your UI
    - Otherwise, need to brute-force

Passphrases



xkcd comics reproduced under CC license from Randall Munroe. Hat Guy is not amused.

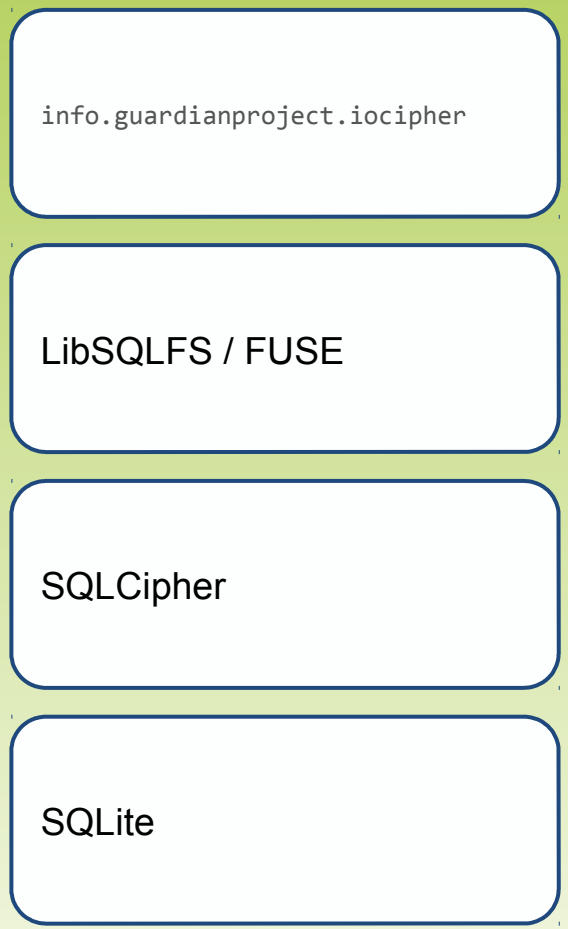
Passphrases

- Multi-Factor Authentication Sources
  - NFC tag
  - QR code
  - Paired Bluetooth device
  - Wearable device app
  - Biometrics (e.g., fingerprint scanner)
  - Formal, fancy-pants enterprise-grade stuff

IOCipher: Locking Down Files

- Virtual Encrypted Filesystem
  - Backed by a SQLCipher database
  - Replacement classes for java.io.File, etc.
  - Feel of working with files, with transparent encryption
  - Designed for user-known passphrases
    - Facebook Conceal: generated passphrase, but only makes external storage as secure as internal storage

IOCipher Stack



Java/JNI wrapper API

Virtual Filesystem that maps to SQL schema / structured database

Encryption layer for SQLite

Base storage mechanism

Moving to IOCipher

- Connect to your encrypted disk's file using `new VirtualFileSystem(dbFile)`
- Mount it with a password using `VirtualFileSystem.mount(password)`
- Replace the relevant import statements:
  - `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 info.guardianproject.iocipher.File;

import info.guardianproject.iocipher.FileOutputStream;

import info.guardianproject.iocipher.VirtualFileSystem;


File dbFile = new File(getFilesDir(), "/myfiles.db");
VirtualFileSystem vfs = new VirtualFileSystem(dbFile);


vfs.mount(yourPasswordFromTheUserGoesHere);


File file = new File(dirPath);

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


NetCipher: Locking Down Teh Interwebs

- Tor Integration
  - Check to see if Orbot is installed, help get it installed
  - Help in proxying HTTPS/SOCKS requests through Orbot
  - Help in accessing Tor Hidden Services

Summary

- Consider Encryption
  - …even if you don't think you need it
- SQLCipher: Easiest Option for Encrypted Database
  - …if you can live with the APK footprint
- IOCipher: Easiest Option for Encrypted "Files"
- NetCipher: Tor FTW!