The Case for Strong Mobile Authentication Biometrics and Mobility: Ready or Not?

A better authentication option

ISC^2 East Bay Chapter

Lee Neely, CISSP, CISA, CISM, CRISC, GMOB, GPEN, CCUV
Senior Cyber Analyst

July 10, 2015

LLNL-PRES-677003
This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

Lawrence Livermore National Laboratory

Introduction

- 30+ Years IT Experience
- UNCLE CU Board Secretary
- Cyber Security Analyst at LLNL
  - CSP New Technology Lead
  - Mobility is my passion - secure ubiquitous access (any time, any device, any place)
- SANS Mentor/Instructor

Introduction

- Mobile Device Administrators and end-users need to be more aware of risks
  - Unauthorized Device Access
  - Data and access granted with physical access to mobile devices
- Developers need to know alternate ways to secure access to applications

Case for Strong Passcode

- Prior to 2007/8, Blackberry was secure Smartphone
  - iPhone: 2007, Android 2008
- First iPhone passcode bypass had a blasé reaction
- 47% users set passcode, gesture or other lock
  - 77% of those use 4 digit pin

The Problem with PINS

- Observable
- Discoverable
  - Social Media
  - User behavior == Easy to recall
  - Most common PINS: http://www.datagenetics.com/blog/september32012/
- Brute Force
  - IP-Box & USB Rubber Ducky < 17 hours

Argument for Stronger Authentication

- Alternate authentication needed that is resistant to discover.
- Authentication that is resistant to brute force
- Authentication that is hard to replicate
- Long & Complex Passcodes can fill this need
  - Users not as welcoming to this

Biometrics

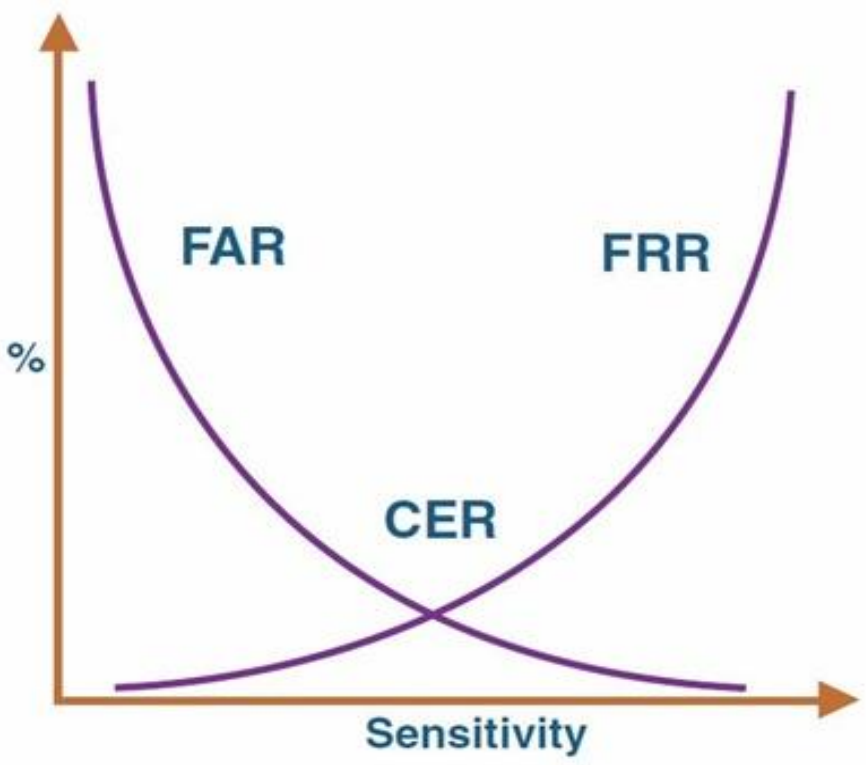
- Authentication based on some intrinsic characteristic of the user
  - Fingerprints
  - Retina scans
  - Face Recognition
  - Voice Prints
- Offsets impact of a strong passcode, and not readily observable

Biometric Evolution

- Android OS 4.1 introduced Facial Recognition
  - Early versions allowed a photograph to authenticate
  - Blinking requirement added – nominally better

- Fingerprint readers
  - Press
    - iPhone 5S
    - Samsung S6
  - Swipe
    - Samsung S4, S5, HTC

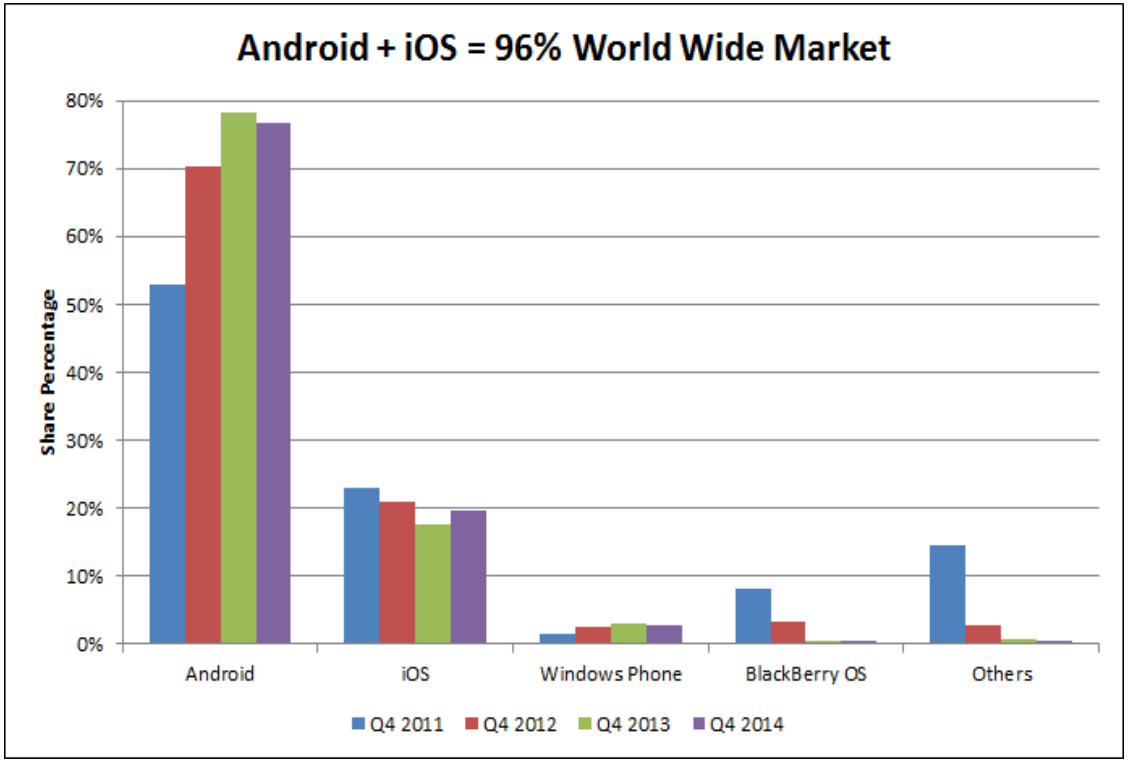
Biometric Sensitivity



Why Android/iOS

- According to data from International Data Corporation (IDC) in Q4 of 2014, 96% of the world-wide smartphone market is comprised of Android and iOS devices.
- Given the substantial margin over other options, I am going to focus on iOS and Android device solutions.

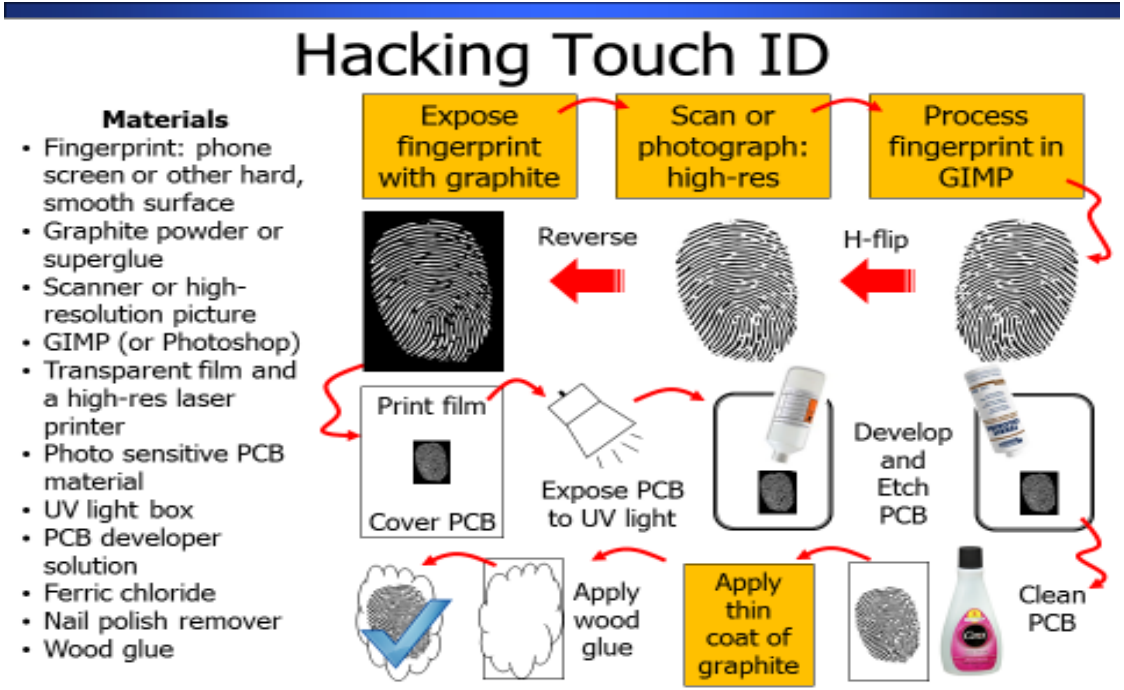
Android/iOS Market Share



Current Solutions

- iPhone 5s ,Galaxy S5, HTC One introduced fingerprint readers
- Bypass techniques using fake fingerprints quickly emerged
- iPhone 6 and Galaxy S6 more mature readers
  - Harder to bypass

Hacking Touch ID



SANS Institute SEC575: Mobile Device Security and Ethical Hacking author Joshua Wright illustrates the CCC process for creating a fake fingerprint to trick Touch ID, as shown in this course excerpt

Protection of Biometric Information

- Most important aspect of Biometric authentication
- Samsung & Apple store a mathematical representation of the fingerprint
- Neither allows fingerprint database to backed up or stored in the cloud

Protection of Biometric Information

- Researchers from FireEye found issues
  - HTC One Max fingerprint database world-read/write
  - HTC One Max fingerprint database not encrypted
  - Fingerprint scanner can be accessed from host OS

Samsung Fingerprint Scan Data Security

- Actual fingerprints or biometric data is not stored. A hash is created from the scan and the resulting hash is stored in Trustzone which is the ARM architecture TEE (Trusted Execution Environment)
- Fingerprint scanner & UI are in TEE
- Fingerprints cannot be accessed outside the TEE
- Fingerprint scanner hardware cannot be accessed outside TEE

Samsung Fingerprint Scan Data Security

- Scanner is connected such that only TEE can access it physically
- TEE takes over display to show trusted UI for input
- Fingerprint data is not accessible outside TEE
- Trustlet provides results of scan, possibly some key protected by successful scan, but no scan information

Apple Touch ID Fingerprint Data Security

- Fingerprint representations in Secure Enclave
- Secure Enclave is its own coprocessor with separate micro-kernel.
  - Even if main kernel is compromised, Secure Enclave still secure

Biometric Impacts on Daily Use

Scenario	Touch ID	Fingerprint Scanner
Reboot/Power Cycle	Must Enter	Only required if on device encryption enabled
Idle more than 48 hours	Must Enter	Can still use fingerprint
After five unsuccessful attempts to match a fingerprint	Must Enter	Must Enter. This is new with Samsung's S6.Previously, no limit was set.
Enrolling/managing fingerprints	Must Enter. Must have passcode prior to enrolling. Maximum five fingerprints stored	Either may be used. Must have a minimum of six character "Backup Password", including one digit and one number configured. Maximum four fingerprints stored
Device receives a remote lock command	Must Enter Passcode	Must use configured lock password. Note remote lock changes the screen lock to password instead of fingerprint.

Piece Of Mind

- Remember time involved in creating a fake fingerprint
- Back it up with a complex passcode
- Other mitigations
  - On-Device encryption: Android – manual enable, iOS – always on
  - Find My iPhone
  - Android Device Manager
  - Auto wipe after preconfigured passcode limit
- Protect the device. Treat it like your wallet

Secure Environments

- Apple: Secure Enclave
- Android: Trusted Execution Environment
- Functionally Synonymous

Background: Global Platform

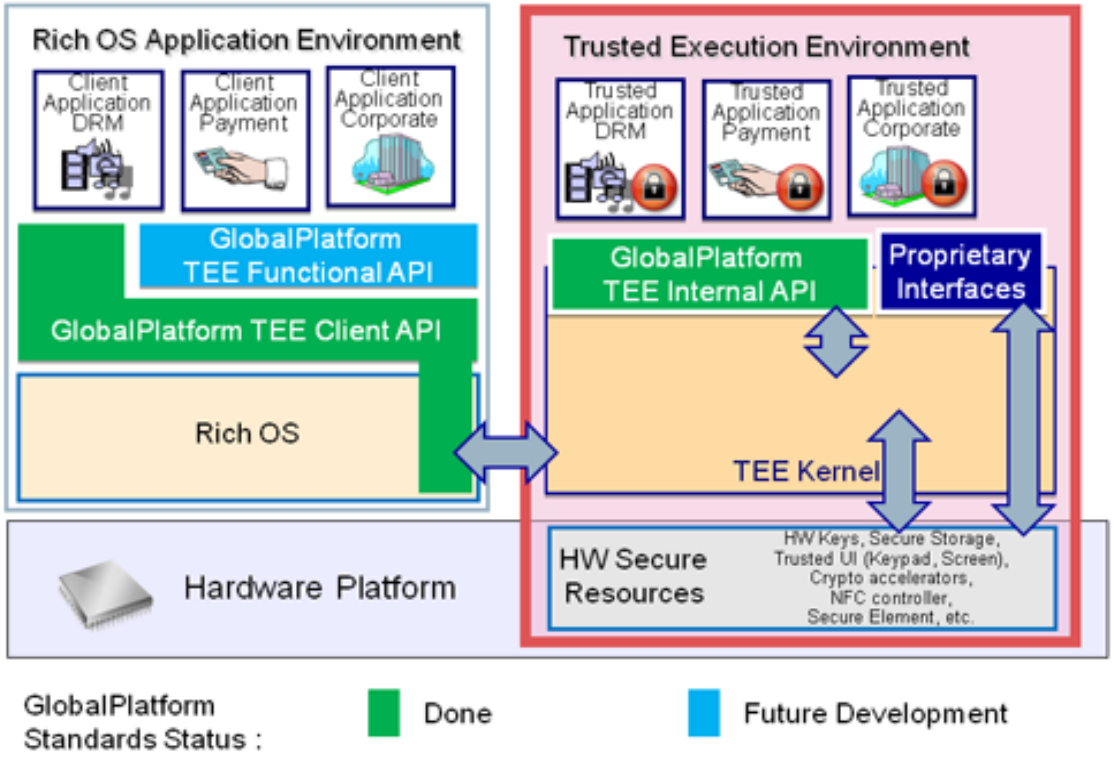


Image from www.arm.com, describes the TrustZone standard

Apple Implementation Details

- Introduced the Secure Enclave with the release of the iPhone 5s/A7 processor
- Uses Micro-Kernel based on L4 Family
- Secure boot process insures integrity of Secure Enclave
  - Integrity check failure forces device into DFU mode
- Part of device memory is encrypted for Secure Enclave use only

Using Touch ID with Applications

- System provided APIs to authenticate with Touch ID
- Keychain items can also be protected with Touch ID
- Process pretty simple using Local Authentication Framework

iOS Local Authentication Framework

Classes

The Local Authentication framework provides facilities for requesting authentication from users with specified security policies. For example, to request that the user authenticate using Touch ID you would use code such as this:

```
LAContext *myContext = [[LAContext alloc] init];
NSError *authError = nil;
NSString *myLocalizedReasonString = <#String explaining why app needs authentication#>;

if ([myContext canEvaluatePolicy:LAPolicyDeviceOwnerAuthenticationWithBiometrics error:&authError]) {
    [myContext evaluatePolicy:LAPolicyDeviceOwnerAuthenticationWithBiometrics
                localizedReason:myLocalizedReasonString
                          reply:^(BOOL success, NSError *error) {
            if (success) {
                // User authenticated successfully, take appropriate action
            } else {
                // User did not authenticate successfully, look at error and take appropriate action
            }
        }];
} else {
    // Could not evaluate policy; look at authError and present an appropriate message to user
}
```

The localized string you present to the user should provide a clear reason for why you are requesting they authenticate themselves, and what action you will be taking based on that authentication.

Classes

NSObject	`NSObject` is the root class of most Objective-C class hierarchies.
LAContext	Represents an authentication context.

Samsung Implementation Details

- Samsung introduced their Trusted Execution Environment in the Galaxy SIII.
- Samsung uses a micro-kernel named *Mobicore*
  - Developed by Giesecke & Devrient GmbH (G&D)
  - Uses TrustZone security extension of ARM processors to create the TEE
- Secure applications that run inside *Mobicore* are called trustlets.
  - Applications communicate with trustlets through the *Mobicore* library, service and device drivers.
- While third-party application developers can create their own trustlets, they need to be incorporated into *Mobicore* by G&D.

Mobicore Architecture

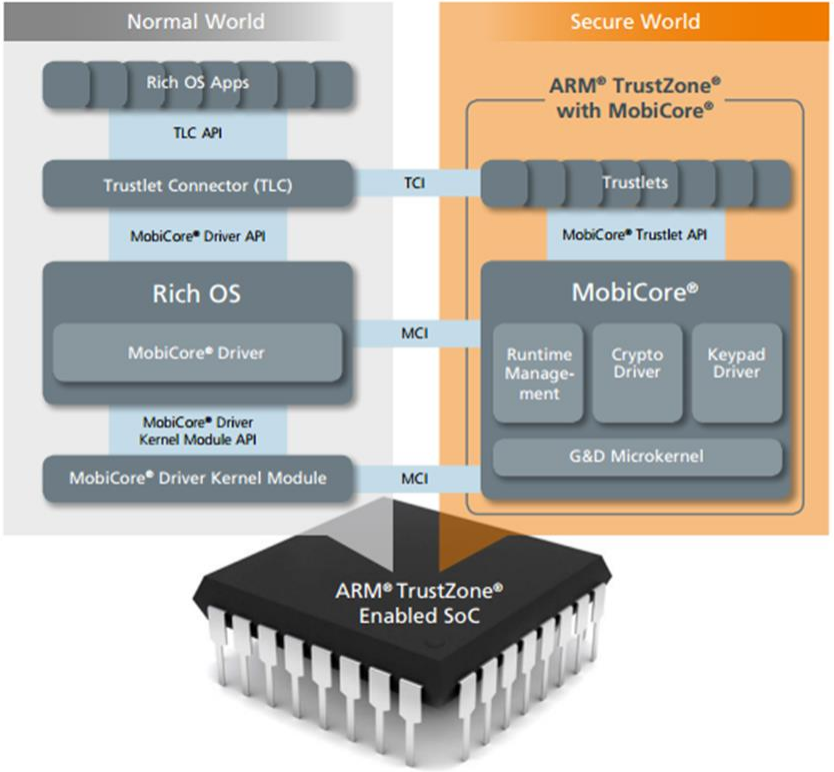


Figure published by G&D

Samsung Fingerprint Scanner use with Apps

Application workflow for Fingerprint Authorization is as follows:

1. User opens app (that uses fingerprint)
2. User goes to authorize something (such as payment)
3. App launches fingerprint Trustlet
4. Secure screen takes over and provides UI for reading fingerprint
5. Fingerprint scanner accepts input
6. Trustlet verifies fingerprint by comparing it to stored scan data from registration and provides authorization back to App
7. Authorization is completed

Samsung API

- To access fingerprints on Samsung, you need to use the Pass SDK. It works with both the Swipe (S5, Note 4) and Touch (S6) sensor

- The Pass SDK provides for not only checking a fingerprint, but also has mechanisms to add, check, and even throw a handled exception in the event your application is run on a device that doesn't support a fingerprint scanner

Sample Code to Authenticate with Fingerprint

```
listeners.put(R.id.buttonShowIdentifyDialogWithPW, new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        logClear();
        try{
            if (!mSpassFingerprint.hasRegisteredFinger()) {
                log("Please register finger first");
            } else {
                if (onReadyIdentify == false) {
                    onReadyIdentify = true;
                    try {
                        mSpassFingerprint.startIdentifyWithDialog(SampleActivity.this, listener, true);
                        log("Please identify finger to verify you");
                    } catch (IllegalStateException e) {
                        onReadyIdentify = false;
                        log("Exception: " + e);
                    }
                } else {
                    log("Please cancel Identify first");
                }
            }
        } catch (UnsupportedOperationException e){
            log("Fingerprint Service is not supported in the device");
        }
    }
});
```

Compromising

- The first option is to directly attack the fingerprint database.
  - The FireEye team found HTC was storing the fingerprint database as a world-readable world-writable file,
- The second option is to interfere with the communication between the fingerprint reader and the SW
  - Again, the FireEye team found cases where this was possible, and the vendors issued patches.
- The third option is to modify the device driver that between the NW and SW portions of the device.
  - Potentially altering the behavior of the fingerprint scan checks to always return true.
- Di Shen's presentation from BH-15 shows how an insecure implementation of the TEE OS and driver in the Huawei devices with the Huawie HiSilicon chipset can be used to alter trusted memory, injecting shell code, accessing fingerprint data, and otherwise compromising the TEE. These compromises do require kernel (rooted) access to be successful.

Mitigating Compromise Risk

- A common thread in addressing these weaknesses is patches or updates.
  - Android users have had a struggle getting updates consistently and in a timely fashion.
  - Google announced monthly Over the Air (OTA) security updates for their Nexus devices
  - Samsung announced a fast track OTA security update process

- An important thing to note is the backward compatibility of the updates.
  - Typically Android devices only have updates for eighteen months, if at all.
  - By contrast, Apple iOS 8.4.1 is still supported on the four-year-old iPhone 4s.

Staying Informed

- Subscribe to data feeds that carry announcements of updates and fixes.
  - Google recently created an Android Security Updates Google Group.
  - Apple announces their updates through their security page and security-announce mailing list.

Summary

- Biometric authentication advantages
  - Mechanism that cannot be merely observed to circumvent
  - The authentication information itself is securely stored
  - Applications call interface through a secure API designed to detect tampering
  - Applications are no longer managing passwords

Summary

- A risk based decision has to be made as to whether to trust biometrics
  - Being aware of the physical security of the mobile device is paramount
  - Protect it like you would do your wallet

- Incorporation into applications allows multi-level authentication
  - Device & Application
  - Or multiple authentications when accessing a given application
  - Remember user acceptance

Conclusion

- In short, adding support for biometric authentication is a nominal impact to in-house developed applications
- Making a pitch for using when developing applications should be an easy option for management to accept
- Seriously consider it for your next mobile application development project

Questions?

Lee Neely
CISSP, CISA, CISM, CRISC, GMOB, GPEN, CCUV
neely1@llnl.gov
@lelandneely

References

- Apple iOS Security Guide: https://www.apple.com/business/docs/iOS_Security_Guide.pdf
- Brute Force Android PIN: http://www.bbrotherton.com/main/androidpinbruteforce
- Samsung Fingerprint 4.4: http://www.samsung.com/us/support/howtoguide/N0000011/16610/225807
- Samsung S5 Fingerprint Hacked: http://www.tomsguide.com/us/samsung-galaxy-s5-fingerprint-hack,news-18655.html
- https://www.pentestpartners.com/blog/brute-forcing-android-pins-or-why-you-should-never-enable-adb-part-2/
- Samsung/Apple face-off: http://www.cio.com/article/2454883/consumer-technology/fingerprint-faceoff-apple-touch-id-vs-samsung-finger-scanner.html

References

- Improvements to iPhone 6 Fingerprint scanner: http://9to5mac.com/2014/09/24/hack-test-shows-apple-improved-security-and-reliability-of-still-not-perfect-touch-id-sensor-in-iphone-6/
- CCC how-to make a fake fingerprint: http://dasalte.ccc.de/biometrie/fingeradbdruck_kopieren.en
- FAR/FRR Graph: http://www.securitysales.com/article/knowing-how-biometrics-can-be-beaten-helps-you-win
- Entrust Blog: http://www.entrust.com/bypassing-fingerprint-biometrics-nothing-new/
- iOS Authenticate with Touch ID: https://developer.apple.com/library/ios/documentation/LocalAuthentication/Reference/LocalAuthentication_Framework/
- Samsung Pass SDK: http://developer.samsung.com/galaxy#pass

References

- Samsung KNOX Security Overview: https://www.samsungknox.com/en/system/files/whitepaper/files/An%20Overview%20of%20the%20Samsung%20KNOX%20Platform_V1.11.pdf

- MobiCore description: https://www.sensepost.com/blog/2013/a-software-level-analysis-of-trustzone-os-and-trustlets-in-samsung-galaxy-phone/

  BlackHat Presentations
  - Fingerprint Weaknesses https://www.blackhat.com/docs/us-15/materials/us-15-Zhang-Fingerprints-On-Mobile-Devices-Abusing-And-Leaking-wp.pdf
  - Attack TrustZone: https://www.blackhat.com/docs/us-15/materials/us-15-Shen-Attacking-Your-Trusted-Core-Exploiting-Trustzone-On-Android-wp.pdf