Whitepaper on identity solutions for mobile devices

How software and hardware features of modern mobile devices can improve the security and user experience of your software

Author: Jonas Lindstrøm

August 2016

ALEXANDRA INSTITUTE

Introduction

The world of mobile devices is very fragmented and deciding what platforms to target is an important part of planning a software development project.

The possibilities include developing a native app running on one or more of the common platforms or developing a web application in JavaScript that can run in the mobile device's browser.

This whitepaper is meant as a guide for software developers who develop software for mobile devices that involves some identity handling. Especially if the software running on the mobile device is rich, meaning that it has to do some or all of the identity handling on its own.

In particular, we discuss the availability on mobile platforms of certain software and hardware features that are useful in such software, and how they can be used to enhance the security and usability of your application.

Is this relevant for me?

This whitepaper is especially relevant for you if you are developing software for mobile devices that are to do cryptographic operations on the mobile device, or if you plan to use hardware features such as secure storage or NFC.

Features we consider

- Software libraries for cryptographic operations and for integer arithmetic with arbitrary precision,
- Embedded secure elements for secure hardware-backed storage of sensitive data,
- NFC capabilities for communication with other devices such as other mobile devices, NFC tags, smart cards, card readers, etc.

These features have been chosen because it turns out that they are all features that can potentially improve both the usability and security of an application, and because the availability of all of them differs from platform to platform. In this whitepaper we consider iOS, Android and Windows Phone which, combined, account for about 95% of the European market, see figure 1.

Example solutions

For inspiration we now present three different existing software solutions for mobile devices that all involve some amount of identity management, and our presentation focus on what software and hardware features of the mobile platform they depend on.

NemID

NemID[1] is the official Danish eID solution. Until recently it was not possible for a user to use it on a mobile device since the client was implemented as a Java Applet running in a browser. This has now been replaced with a JavaScript solution that makes NemID usable on mobile devices[2].

The user has a secret password as well as a paper card with one-time-passwords. To authenticate, both usually have to be used, but in some use cases it is also possible to authenticate without a one-time-password, which gives the user a limited access. However, the user does not reveal his password towards the identity provider but uses a cryptographic protocol to prove towards the identity provider that the password he entered is the right one without revealing it[3].
To do this, it must be possible for the client application to perform cryptographic calculations on behalf of the user, hence the need for a Java Applet or a JavaScript application running in the user's browser. This also require libraries for cryptographic operations and for integer arithmetic with arbitrarily large integers on whatever platform the client is running on.

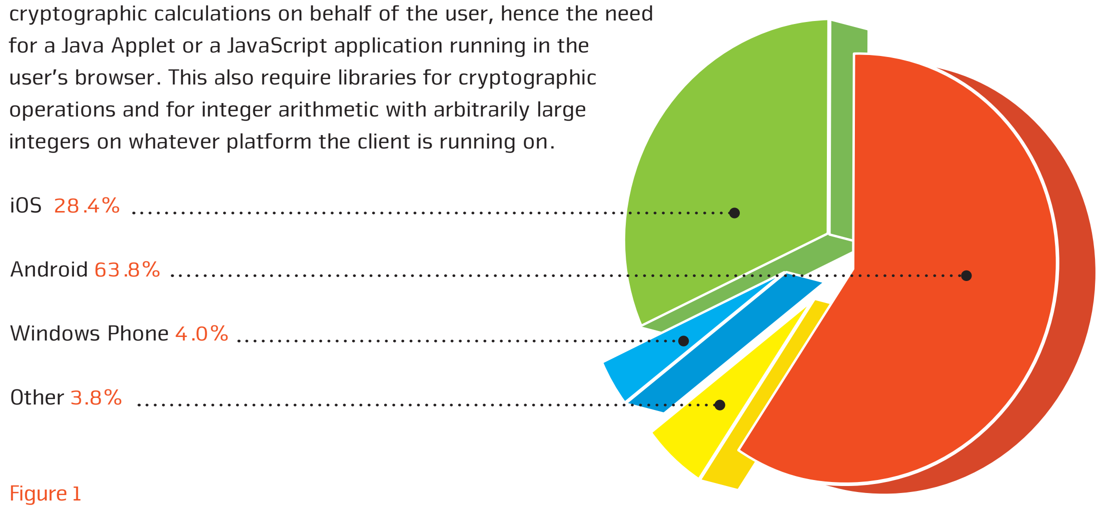

Figure 1
Market shares of common mobile operating systems in Europe as of june 2016.
See http://stats.areppim.com/stats/stats_mobiosxtime_eu.htm.

1 *https://www.nemid.nu/dk-en/*
2 *http://www.nets.eu/products/nemid-javascript/about-nemid-javascript/Pages/default.aspx*
3 *https://www.nemid.nu/dk-da/om_nemid/sikkerhed/teknikken_bag_nemid/ (in danish)*

ABC4Trust

The ABC4Trust EU research project[4] works to promote privacy-preserving attribute-based credentials, which is a cryptographic technology that enables a user to control what information from his credential he wants to reveal, and especially what information he does not want to reveal, towards a service provider (e.g. only his age but not his name).

Part of the project has been the development of a reference implementation that has been tested in two pilot projects. In the pilot projects the user accesses services from a web browser running on a laptop. The user's credentials are stored on a smart card, which also has to be present in order for him to use his credentials. This kind of hardware-backed security is desired because it ensures that credentials cannot be shared with other users and that a malicious adversary cannot easily extract and use a user's credentials.

The user needs to run an application on his laptop, which handles some cryptographic operations and the communication with the smart card, while the web browser communicates with the application via a browser plugin.

The reference implementation has been ported to the Android platform, allowing the user to use his credentials from a smart phone. However, browser plugins are not available on mobile platforms so the architecture from the pilot projects could not be ported directly to mobile platforms. Instead it was implemented as a native Android app.

MobilePay

MobilePay [5] is a payment platform for smart phones which is developed by Danske Bank. Using this solution, users can make payments from their smart phone directly to other users, to shops or to smart phone apps. This is done using an app, MobilePay, which is available for iOS, Android and Windows Phone.

When registering as a user of MobilePay, the user registers a credit card and a bank account with his phone number. Now, when paying user-to-user, the paying user enters the phone number of the receiver, and the bank arranges a credit card transfer from the paying user to the account of the receiving user.

When using the app in a store, the user either presents his or her phone to a reader attached to the cash register, which using NFC or Bluetooth communicates with the users' smart phone, and the payment is completed upon confirmation from the user in the app. If NFC or Bluetooth is not available, the user can read a QR-code which is printed on the reader using the MobilePay app.

The app also allows the user to make payments to other apps running on the users' smart phone using a solution called MobilePay AppSwitch. An API is available for developers wishing to use this feature.

4 *https://abc4trust.eu/*

5 *http://www.mobilepay.dk*

Software features

For any software development project, the choice of what software libraries to use is important. The choice is further complicated when third-party libraries are necessary, because here we need to not only consider the capabilities and functionality of the library but also the licence conditions.

What about performance?

JavaScript code executed in a browser does not perform as well as native applications. For most use cases this is not a problem on modern-day devices, but cryptographic operations, especially public key cryptography, can be very computationally demanding.
So watch out if you are doing a lot of such operations!

Basic cryptographic operations

Most identity solutions, and in general any software which has to deal with security of some sort, will need to be able to perform basic cryptographic operations such as cryptographic signatures and encryption using common cryptographic technologies, e.g. RSA, and hashing with state-of-the-art algorithms, such as e.g. SHA-256, and using good implementations of them are crucial to the security of the solutions.

Libraries with such cryptographic functionality are available on all native platforms. For JavaScript, W3C is developing a Web Cryptography API[6] intended to be implemented in web browsers and to be available to JavaScript applications running in the browser. However, only the random number generator from this API has so far been widely implemented. Note that this could change in the near future, so it is advised to consult the documentation for the target browser's platform or the Web Cryptography API.

It is also possible to use third-party JavaScript libraries with Google's End-to-end[7] and Microsoft Research's JavaScript Cryptography Library[8] being the most notable candidates. Both libraries have permissive licenses, allowing developers to use them for free as part of both commercial and non-commercial software.

One possible issue with doing cryptographic operations in JavaScript is that applications written in JavaScript do not perform as well as native applications, which could be critical, especially on devices with less computational power, such as e.g. mobile devices. Furthermore, we have conducted some benchmarks, which show that performance varies greatly from platform to platform; so if you plan to develop a performance critical application in JavaScript, do keep this in mind. Our experience shows that the slowest platforms are Internet Explorer on laptops and Safari on mobile devices. However, this may vary depending on the library and what operations are being done.

Arbitrary precision integer arithmetic

For most use cases a good crypto-library, as discussed above, will be enough. Due to the risk of making security critical errors, it is usually not recommended to implement such cryptographic functionality on your own, also known as don't-do-your-own-crypto. But if you are developing software where you have to implement custom cryptographic operations, e.g. if you are working on a cryptographic solution, you will likely need to be able to do arithmetic with arbitrarily large integers. As with the cryptographic libraries, it is important to choose a good, well-implemented library, both due to security issues caused by implementation errors such as e.g. bad memory handling which may leak information, but also for performance reasons.

Both Android and Windows Phone offer libraries for such calculations, in both cases named BigInteger, but on Apple iOS you will need to find a third-party library, e.g. one of the many existing C or C++ libraries such as GMP[9], released under the GPL-license, or OpenSSL[10] released under the Apache License. There are also

6 *http://www.w3.org/TR/WebCryptoAPI/*

7 *https://code.google.com/p/end-to-end/*

8 *http://research.microsoft.com/en-us/downloads/29f9385d-da4c-479a-b2ea-2a7bb335d727/*

9 *https://gmplib.org/*

10 *https://www.openssl.org/docs/crypto/bn.html*

Objective-C libraries such as Æquens[11], both released under the MIT-license.

In JavaScript, both Google's End-to-end and Microsoft Research's JavaScript Cryptography Library provides functionality for arbitrary precision integer arithmetic. Note that the discussion provided on JavaScript performance given in the preceding section is also relevant here.

Are the APIs any good?

Unless you are implementing your own cryptographic protocols, the available APIs for accessing the secure element of mobile devices have the functionality needed for most use cases. This includes encrypting, signing and verifying data using a key stored on the secure element.

Hardware features

Hardware-backed storage

Sensitive data stored in the memory of a mobile device is vulnerable to a number of attacks: Malware or an adversary with physical access to the device could extract the data, violate the user's privacy and possibly use the data to impersonate the user towards a third party. Some mobile platforms offer storage of a user's private keys on an embedded hardware token, a secure element which is able to perform cryptographic operations involving the user's private key without the key ever leaving the chip. This makes it hard for an adversary to extract the key and use it on behalf of the user. Typically, the secure element is also protected by a password or PIN code.

Recent versions of Android offer hardware-backed storage of keys on devices that have a secure element, which is the case for most newer devices. The feature has been used by Google in Google Wallet for a while, but it has not been available through the development API until Android 4.3. In order to use the secure element on Windows Phone, you will need a special permission applied to your developer account, and once this permission is given, you can send arbitrary commands to the secure element. Only Apple's newest iPhone, the iPhone 6, contains a secure element, and there is not yet any public API available.

In JavaScript it is currently not possible to communicate with secure elements. The W3C group has discussed having support for smart cards, but it is uncertain whether and when this will be available to developers.

Public APIs

For both Android and Windows Phone, the API of the secure element offers a limited list of operations that can be performed with the key - consult the APIs[12] for details. For custom applications that need other features than the ones provided by the API, a possibility to achieve hardware-backed security might be to use smart card for storage and use the mobile device's NFC capabilities to communicate with the smart card that has to be held against the back of the mobile device while in use (see the next section for a discussion on NFC). Using a smart card with a mobile device might be a bit cumbersome, but it could be an option in some use-cases because smart cards can be equipped with custom applications, and are easier to deploy than e.g. SIM cards.

For Android devices there are a number of alternative operating systems, most notably CyanogenMod[13], which has a more permissive API for interacting with the secure element (and for a number of other features too, e.g. NFC). This will only be relevant in certain use cases, e.g. if there are very few users, since it requires a user to install the alternative operating system on his device.

11 *http://aequans.com/en/biginteger*

12 *http://developer.android.com/guide/topics/connectivity/nfc/index.html and http://msdn.microsoft.com/en-us/library/windows/apps/microsoft.phone.secureelement%28v=vs.105%29.aspx*

13 *http://www.cyanogenmod.org/*

Platform availability	iOS	Android	Windows Phone	JavaScript
Basic cryptographic operations	🟩	🟩	🟩	🟨 1
Arbitrary precision integer arithmetic	🟨 1	🟩	🟩	🟨 1
Hardware-backed storage	🟥 4	🟩 3	🟩 2, 3	🟥
Communication via NFC	🟥 4	🟩 3	🟩 3	🟥

1 Only available through third-party libraries.
2 A special permission from Microsoft is needed.
3 This feature is available on most devices.
4 Only available on the newest iPhone 6 and there is not yet any API.

NFC

Near-field communication (NFC) is a technology that is used by mobile devices to communicate with NFC-tags, smart cards, smart card readers and other devices that are near (< 0.2 m) the mobile device. This feature is useful if the mobile device is part of a larger setup and has to communicate with other devices in its proximity. An example of this is Google Wallet, but you might also imagine other setups, e.g. where a mobile device is used as an authentication token[14]. Using the NFC-capabilities of a phone in a solution can give a very smooth and intuitive user experience, since the only thing a user has to do to communicate with another device is to swipe his phone across a reader.

NFC has only recently been adapted by Apple, and is available on the most recent iPhone 6, but for now it will only be used for Apple's payment system, Apple Pay and will not be available to developers. Both Android and Windows Phones do have NFC capabilities (at least the most recent devices) and both platforms also have APIs for utilizing these capabilities in different ways: An NFC-enabled mobile device can emulate a smart card, known as Host card emulation, which has been available on Android since version 4.4 and also on Windows Phone. It can also be used to communicate with a smart card, which is also available for both Android and Windows Phone – it is even possible to send custom commands to the card, which is useful e.g. if the application needs to communicate with a custom smart card application.

14 *For a detailed discussion on how to use the NFC capabilities of a smart phone for identification and authorization, e.g. as an access token, see Smart Card Alliance's whitepaper Mobile Devices and Identity Applications: http://www.smartcardalliance.org/publications-mobile-devices-and-identity-applications/*

Conclusion

There are many things to consider when planning a software development project on mobile devices. The world of mobile devices is very fragmented, and deciding what platform(s) to target (web applications, iOS, Android, Windows Phone, etc.) is crucial. The choice affects not only what potential users the software will have. It also affects what software and hardware features are available to the software. So a developer needs to carefully consider whether the features needed for a specific project are available on the platforms he intend to target.

Where should I go for more?

If you are planning to use one of the standard web-based protocols for identity managenment, such as Facebook Connect, OAuth, OpenID connect or UMA, the Alexandra Institute has published a whitepaper called 'A short guide to B2C authentication standards' that might be useful.

See the downloads section of: http://www.alexandra.dk/dk/labs/security-lab/sider/security-lab.aspx

In this whitepaper we have discussed the availability of certain software and hardware features that are often used with software that has to perform identity handling: Libraries for cryptography and arbitrary precision integer arithmetic, secure elements for hardware-backed secure storage and NFC. It turns out that the availability depends strongly on the platform, which makes it imperative for a developer to take this into account when planning a project, especially if certain hardware, such as secure elements and NFC chips, has to be used. Many Android and Windows Phone devices have both secure elements and NFC and the platforms have APIs for using them. On iOS, only the most recent device, iPhone 6, has these features, and there is no public API for accessing them.

There are software libraries for cryptography available on all platforms. These should be enough for most use cases, and we recommend that developers use existing libraries. However, if a software project involves implementing custom arithmetic cryptographic algorithms, a library for arbitrary precision integer arithmetic might come in handy, and such libraries are available on Android and Windows Phone. On iOS, only third-party libraries are available, but some of the commonly used libraries are released under the GPL-license, so watch out if the GPL-license is not acceptable for you.

A JavaScript application currently has no access to the hardware features of the device, but there are nice libraries for cryptography and arbitrary precision integer arithemtic (released by Microsoft and Google). However, since cryptography can be computationally intensive, especially when using public key cryptography, and since JavaScript applications do not perform as well as native applications, a developer might have to be careful to make sure that an application performs well enough on all devices and platforms.

We conclude that mobile platforms offer many interesting software and hardware features for software developers to use for enhancing the usability and security of their software. However, if the software has to be usable across several platforms, caution is advised, especially if software relies on certain hardware-specific features.

Happy coding!

RESEARCH DEVELOPMENT INNOVATION VALUE GROWTH WELFARE

The Alexandra Institute is a non-profit company
that provides research, development and innovation within IT.
Our mission is to create growth, welfare and value.

Contact us

Gert Læssøe Mikkelsen
Lab Manager
Security Lab
+45 24 26 99 11
gert.l.mikkelsen@alexandra.dk

THE ALEXANDRA INSTITUTE
Aabogade 34 • DK-8200 Aarhus N • +45 7027 7012
Rued Langgaards Vej 7, 5B • DK-2300 Copenhagen S • +45 7027 7091

ALEXANDRA
INSTITUTE