SCC - Secure Credential Cloning

This document describes an on-line-based system for provisioning PKI and similar objects in mobile devices in a secure and convenient manner with BYOD and consumers as primary target. The provisioned data is referred to as "credentials" throughout the rest of this document.

Preconditions

The described scheme depends on that the user already has an account at the provider in question that is reachable from a PC over the web. Authentication to the account may be performed in different ways including through password-, OTP- and PKI-solutions as defined by the provider's policy.

The exact nature of provisioned credentials may span everything from a "clone" of the identity already associated with the provider account to something like a payment credential (if the provider is a financial institution).

Objectives

The core question is how you can securely use an authentication to the provider performed on a PC for provisioning credentials in *another* device.

Since we tend to have multiple devices the ability to keep track of in *which device* a certain credential is stored is also of importance, which makes a "Device ID" a logical part of such a system.

The "Device ID" used in the scheme outlined in this document, is actually a short-hand version of an *active* identity in the form of an X.509 certificate and associated private key which can optionally also vouch for other qualities such as the brand and type of the cryptographic container which may be needed for *payment credentials*.

On the succeeding pages the process is first described, then followed by a technical discussion.

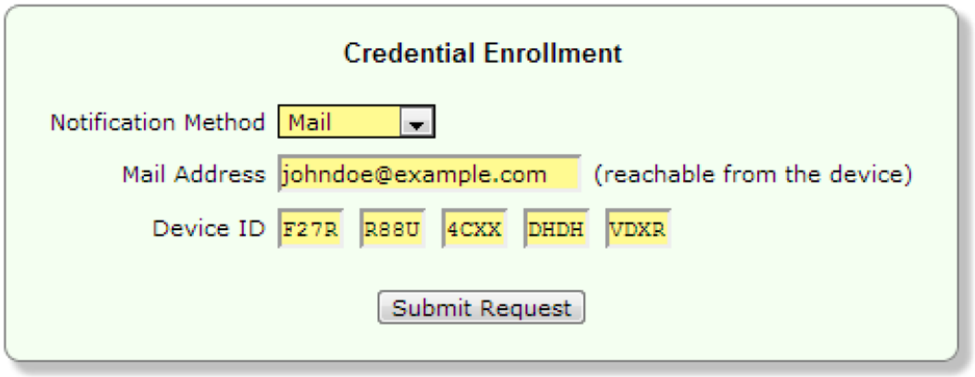
The Enrollment Process from a User Perspective

① The user authenticates to an account

Since this process is outside of the core, it is not shown here…

② Signing up for an enrollment

After successful login, the credential provider asks the user to fill in a web-form like the following:

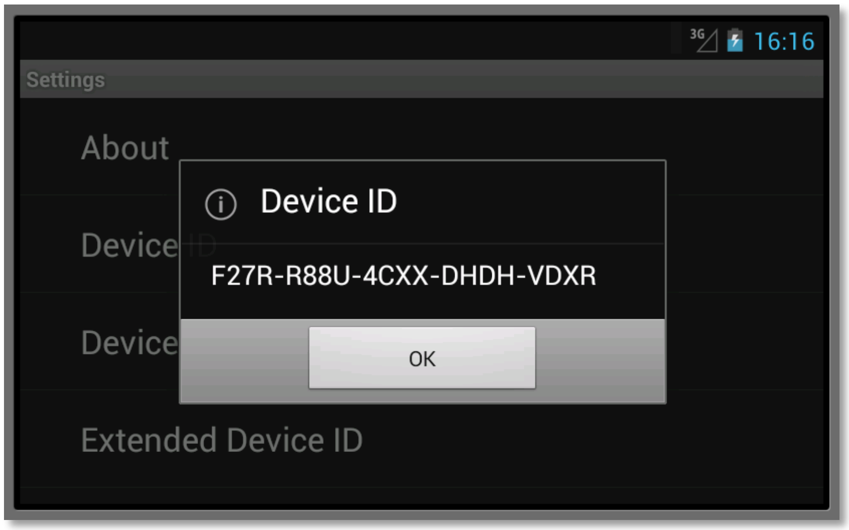


Credential Enrollment

Notification Method	Mail
Mail Address	johndoe@example.com (reachable from the device)
Device ID	F27R R88U 4CXX DHDH VDXR

The Notification Method tells how an **Enrollment URL** is to be delivered to the client device. Mail and SMS are the most common alternatives. Since many mobile phones have a rear camera as well, providing and invoking the URL through a **QR-Code** is also a viable option.

The **Device ID** to be typed into the form is to be found in a system menu of the mobile device:



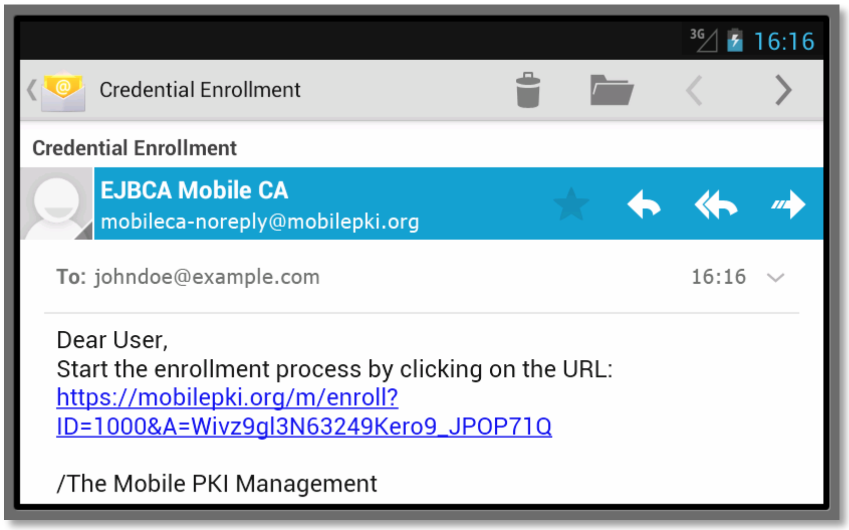
Device ID

F27R-R88U-4CXX-DHDH-VDXR

Continued on the next page…

③ Receiving the notification in the mobile device

A few seconds after the submit operation in the web application, the mobile device will indicate that there is a new message which the user finds by opening the message application:



Credential Enrollment

EJBCA Mobile CA
mobileca-noreply@mobilepki.org

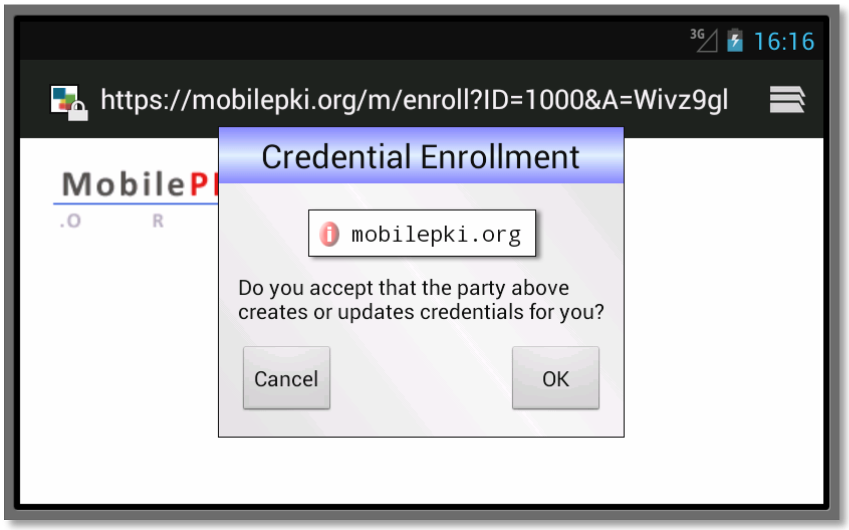
To: johndoe@example.com 16:16

Dear User,
Start the enrollment process by clicking on the URL:
https://mobilepki.org/m/enroll?ID=1000&A=Wivz9gl3N63249Kero9_JPOP71Q

/The Mobile PKI Management

The message contains the **Enrollment URL**. Clicking the URL will open the browser which in turn will invoke a local process returning a dialog as shown in the next step.

④ Enrollment in the mobile device



https://mobilepki.org/m/enroll?ID=1000&A=Wivz9gl

Credential Enrollment

mobilepki.org

Do you accept that the party above creates or updates credentials for you?

Depending on if credential-specific PIN-codes are to be set or not clicking "OK" may be the only thing left to do in order to enroll the credential(s).

The information icon opens a dialog when pressed that informs the user about SSL certificate details as well as **Privacy Implications** of the **Device ID**.

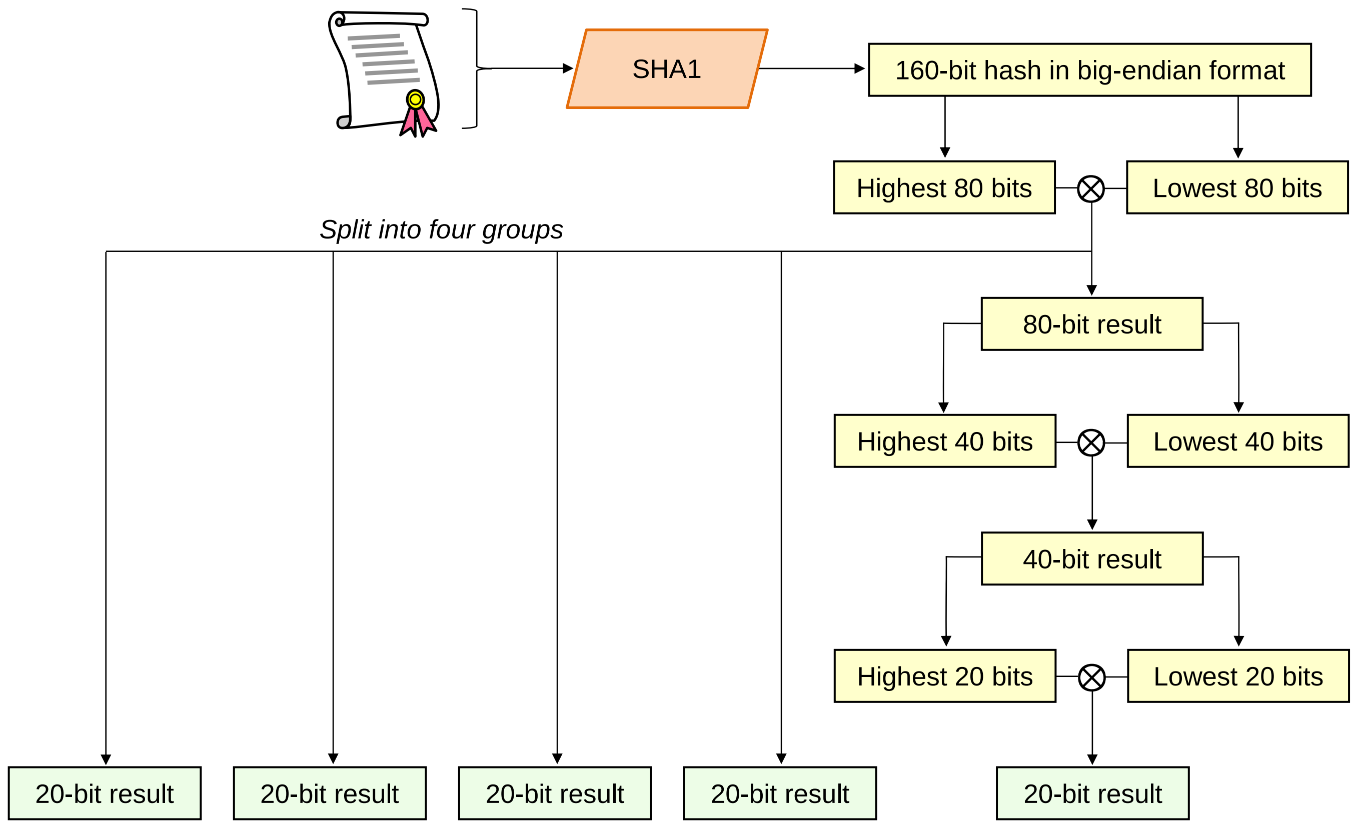
The process typically takes 2-10 seconds and should terminate with a (not shown) web-page holding a success message.

Technical Details

The following pages outline the technical underpinnings of the Secure Credential Clone concept.

Device ID

A Device ID is a textual version of a short hash of an X.509 **Device Certificate** created as follows.



The four result groups plus the fifth (which serves as a checksum) are subsequently converted in to human-readable groups of four characters each, using a modified Base32 character-set: '`ABCDEFGHJKLMNPQRSTUVWXYZ23456789`'.

The checksum makes it less likely that a user manages to fill in an incorrect Device ID since the GUI can immediately detect input errors.

There is also a full SHA1 version of Device ID but since it requires inputting 36 characters, it might not be realistic from a convenience point of view. Also see **Security Considerations**.

Enrollment URL

The Enrollment URL is cryptographically enhanced with a long random value making it difficult to guess. The sample system uses a 162-bit nonce.

The Enrollment URL should preferable not be valid for an excessive amount of time and must also not allow multiple tries.

The Enrollment URL is uniquely associated with a specific enrollment session and user.

Device Certificate

The biggest difference between a **Device ID** and for example an IMEI-code is that the former is a hash of an X.509 certificate which is exchanged during the enrollment process, typically in the form of an attestation signature using the associated private key.

Privacy Implications

Since the **Device Certificate** is a static identity it could potentially be used for tracking purposes. However, the target for the Secure Credential Cloning concept is a limited set of trusted providers such as banks, employers etc.

In reality an email-address is much more powerful as tracker since it is searchable, used in many contexts, globally unique, associated which a specific provider, and usually lasts for a long time, while devices tend to only live a few years.

QR-Code

Using a QR-code instead of mail or SMS is both convenient and secure (since the **Device ID** compensates for possible eavesdropping).

Security Considerations

The described scheme binds a cryptographically strong URL to a device using a *virtual*, *more or less unique*, *device-specific identifier*. User-related enrollment-data is assumed to be associated on the *issuer-side* by the particular **Enrollment URL**.

The actual credential enrollment scheme would typically "counter-sign" a certificate request using the private key of the **Device Certificate** and supply the **Device Certificate** as a part of the request as well.

The provider **must** then (before issuing a credential) create a **Device ID** from the received **Device Certificate** and compare it with the *claimed* **Device ID** received (through the user) during the signup process.

The provider **must** also verify that the received counter-signature matches the public key of the received **Device Certificate**.

Any deviation **must** immediately terminate the enrollment process.

Assuming that the device itself is reasonable secure and provides protection against Internet-based malware attacks the Device ID should be useful as a security mechanism.

It is true that you may be able to create a **Device Certificate** with the same **Device ID** as another device since the entropy is not unlimited. However, to exploit that you need to get hold of a specific **Device ID**, have plenty of computing power at hand, as well as intercepting the device-targeted **Enrollment URL**.

The security of the enrollment process should also be seen in the light of the security available during *usage* of credentials.

Adding enrollment PIN-codes is also a way to improve the binding between the device and an enrollment session.

Long-term, **Device Certificates** should be issued by trusted providers which make attacks of the kind described above essentially impossible, since "home-made" certificates will simply be rejected by the credential issuer.
This would though be meaningless until hardware-protected key-storage becomes the norm.

Implementation

The Secure Credential Clone concept is an intrinsic part of the **https://code.google.com/p/openkeystore** project. In the current incarnation *self-signed* **Device Certificates** are automatically created by the client software itself.

Anders Rundgren, February 2013