

SKS – Secure Key Store

SEs (Security Elements), like Smart Cards, have since decades back been the preferred solution for storing cryptographic keys for authentication etc. This document outlines an enhanced SE-architecture which is *loosely* derived from TrustedComputingGroup's TPM (Trusted Platform Module) 1.2 specification. Due to the fact that SKS unlike TPMs does not imply securing the environment surrounding it, the SKS scheme is fully applicable to existing smart cards (after a firmware upgrade), which is a way reaching the critical mass for a secure key-provisioning facility that is lacking in for example current Internet browsers. A core characteristic of SKS is the ability "emulating" a set of independently issued and managed smart cards using on-line communication only. Although the preferred computing platforms for housing SKS are mobile phones, it should also be quite useful for enhanced USB memory sticks combining mass storage with secure key-containers. Rather than going into details, this document primarily covers the differences between traditional smart cards, TPM 1.2, and SKS.

Key Execution

With key-execution this paper denotes basic signature and encryption functions etc. In this respect all three variants of SEs are essentially identical and can be readily supported by "legacy" cryptographic APIs such as PKCS #11.

Key Protection

PIN-codes are supported by all SEs, but TPMs do currently not have per-key-settable PIN retry-counters. Since SKS was explicitly designed to function as a multi-issuer container, each issuer may deploy their own PUKs (Personal Unlock Keys), as an alternative to an SE device PUK that all issuers can use

Key Management

Key-management refers to deletion, renewal, and cloning of keys and certificates. In contrast to TPMs and smart cards, SKS utilizes PKI-based "proof-of-issuance" methods for isolating issuers from each other making remote updates in a shared container equally secure (but considerably more convenient), as using separate physical tokens, while still not requiring installation of additional trusted roots or similar. By using X.509 certificates as universal key-IDs, the SKS key-management operations can be applied to symmetric keys as well.

Key Provisioning

Key-provisioning is an area where SKS markedly departs from smart cards because even if you buy a \$100 card; it still doesn't enable an *on-line issuer* to verify that keys were actually created in the card! The reason for this is fairly simple: smart card production and personalization is almost exclusively performed in-house by specific staff and software.

Because TPMs were designed to be bolted onto platforms, on-line provisioning became the only realistic method for distributing credentials. To also make on-line provisioning *trustworthy*, TPMs introduced embedded device (platform) keys and device attestations. SKS heavily builds on this concept but does not use the same methods due to its quite different heritage.

When TPMs were conceived the primary target was *more or less anonymous individuals doing business on the Internet while running on provably trusted platforms*. SKS is more like a branch on the European eID tree which means that hiding the device identity is not an option; device identity is probably rather a *prerequisite* (for the *issuer* only NB) since current eID card issuers know exactly which card they have issued to a certain individual including associated credentials.

It doesn't seem entirely unlikely that enterprises and on-line banks also would appreciate having an idea of in which devices they deploy their precious credentials.

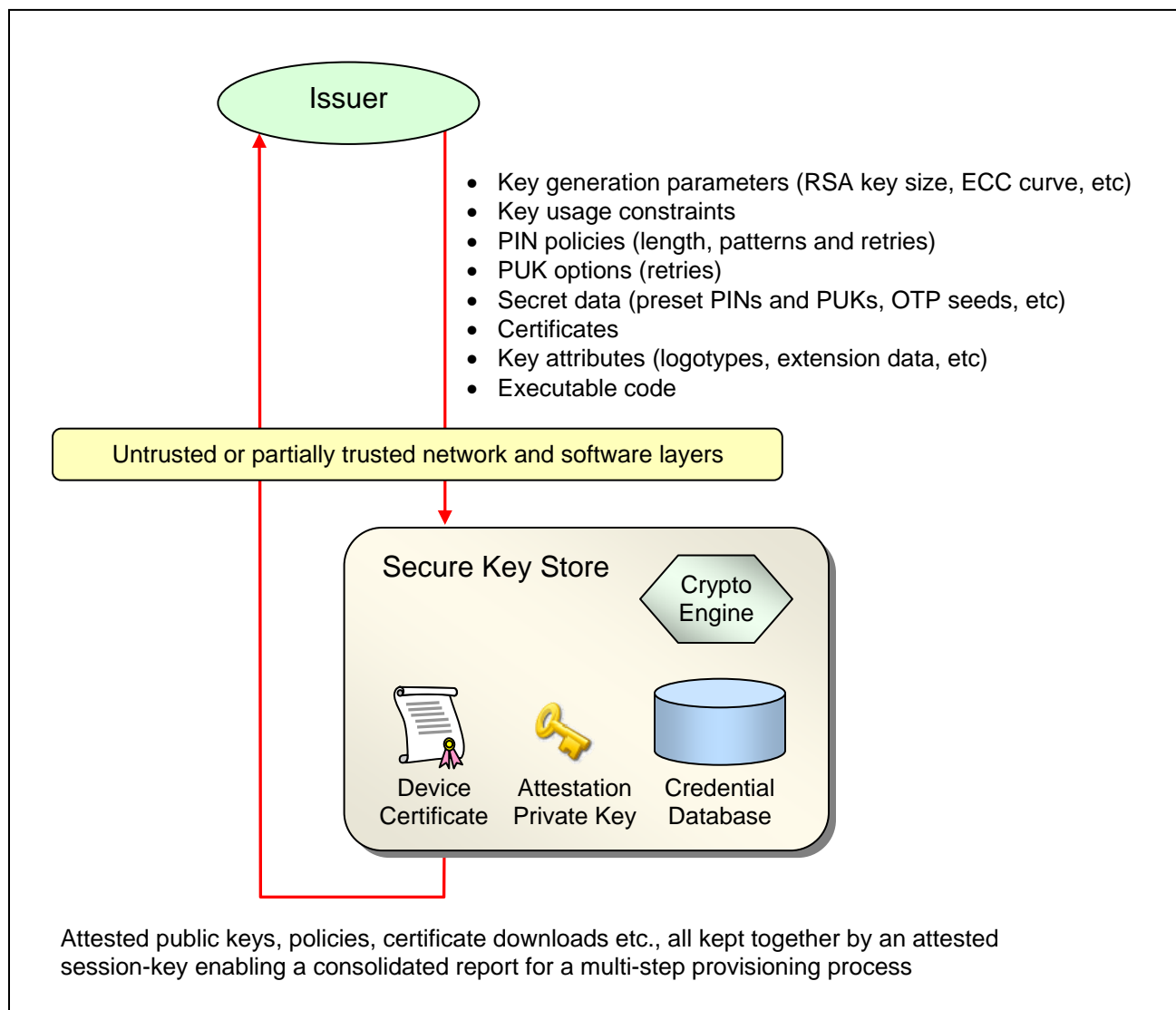
Support for Other Credentials

A distinguishing feature of SKS is that it supports a universal credential extension mechanism which for example can be used for Microsoft's Information Cards. This could also be used for downloading executable code to programmable SEs like JavaCards and .NET cards.

“Air-tight” Key Provisioning

As already mentioned SKS depends on the TPM device attestation concept. However, because SKS was explicitly designed to also be usable in a potentially untrusted computing environment (“unmeasured” using TPM terminology), SKS leverages device attestations for securing *all* aspects of a credential provisioning process ranging from key generation, key usage settings, PIN policy definitions, downloading secret data like PUKs and OTP (One Time Password) seeds, to the final step where certificates are fitted to their respective private keys.

Below is a picture showing the architectural components and a simplified information flow of the air-tight provisioning system supported by SKS.



Dual-use Device IDs

The *Device Certificate* is used as a trusted device ID in a provisioning process. That is, an issuer may reject requests attested by an SKS from an (for the issuer) unknown vendor.

A side-effect of using certificate-based device IDs is that you can create efficient and still quite secure *on-line* enrollment processes where a non-authenticated user signs-up for credentials by sending an SKS-attested request to an issuer. The issuer can then verify the user’s identity with an OOB (Out Of Band) method meeting the issuer’s policy which can be anything from the classic “e-mail roundtrip”, to requiring the user to show up in an office with an identity card. The final part is asking the user for something like the first 8 characters of the 40 hexadecimal-digit SHA1 certificate hash which is the short-form of the device ID. If the answer is matching the device ID of the request, the issuer returns the completed credential package to the user’s SKS. Note that this scheme can completely eliminate enrollment passwords!

Improved methods for remote provisioning are primarily needed for consumers, mobile phones, employees in virtual organizations, as well as for replacement credentials for people who have lost their card far away from the main office.

Provisioning Server Keys

In addition to supporting user-keys, air-tight provisioning also enables organizations getting away from distributing PEM-encoded private keys and password-protected PKCS #12 files on USB-sticks for securing internal servers and Web Service (SOA) clients. That is, an SKS-based key-generation request can neither be moved to another machine, nor requiring any password.

By eventually putting the SKS inside the OS, you would give applications access to private-key operations based on *user*, rather than putting passwords into the applications (like a servlet `web.xml` file). Although server-keys may not be a major security issue, it is at least an administrative improvement knowing exactly which machine that has a certain key.

For externally certified server-keys, the road to SKS-protected keys is likely to be entirely dependent on the outcome for SKS for user-keys. *It appears that a viable option would be including support for plain-vanilla PKCS #10 requests as well.*

Note that servers or other devices using air-tight provisioning do not necessarily need external access to the Internet or to an intranet; off-line provisioning is equally applicable and secure.

Secure “Credential Cloning”

Sometimes there is a need for another copy of a credential like when you have a traditional smart card for use with a PC and would like to use the same credential in a mobile phone. An SKS can be used to securely provision a new credential where the original credential serves as a proof that you already are identified, enrolled etc. This is a cleaner method than exporting keys because 1) it does not compromise SE security. 2) a cloned credential, unlike a copied ditto, due to its unique serial number or similar, allows you to keep track of the actual credential container which is of great importance in the case of device theft. Using web-based on-line methods, credential cloning may be performed by end-users using relatively simple processes, like in an authenticated session based on the original credential, register the device ID of the SKS to be provisioned, and then in another session issue a separate credential request with the new SKS.

Additional Information

For those who are interested in details regarding air-tight provisioning, it is recommended taking a peek at <http://keycenter.webpki.org>

Author:

Anders Rundgren,
anders.rundgren@telia.com, anders@primekey.se