Universal Key Ring – The Time Has Come!

Anders Rundgren, WebPKI.org

This presentation outlines a security architecture, provisioning, and management scheme for secure cryptographic keys, targeting a wide variety of applications including *Virtual SIMs*, *On-line Banking*, *Payments*, *e-Government Access*, and *Enterprise Login*.

In the core of the architecture there is component coined **SKS** (Secure Key Store), which leverages the TEE (possibly aided by a local security processor).

To facilitate easy enrollment of **SKS** keys, a *matching browser-based provisioning protocol called* **KeyGen2** has been developed as well.

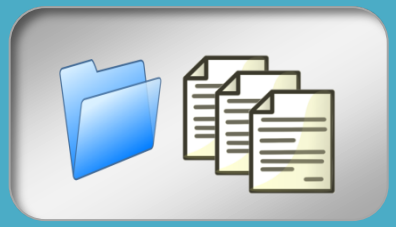
Since cryptographic keys (unlike files), usually represent “relationships” to *external parties,* the scheme provides extensive support for different policies including an ACL system which through the OS/TEE layers, governs which applications a key may be used with.

A side-effect of this arrangement is that cryptographic keys become first-class OS objects like files.

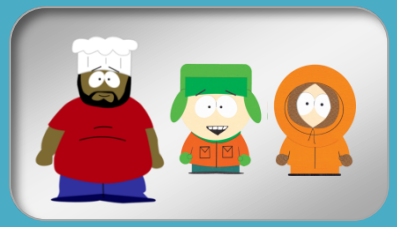
This effort is *complementary* to FIDO alliance. In fact, it seems quite feasible building FIDO alliance products and SKS/KeyGen2 on the very same security platform.

Core OS Objects

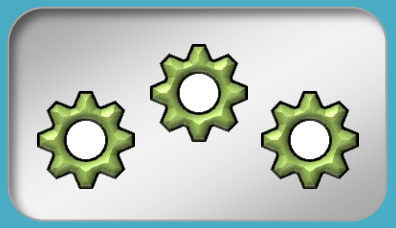
Devices



Files



Users

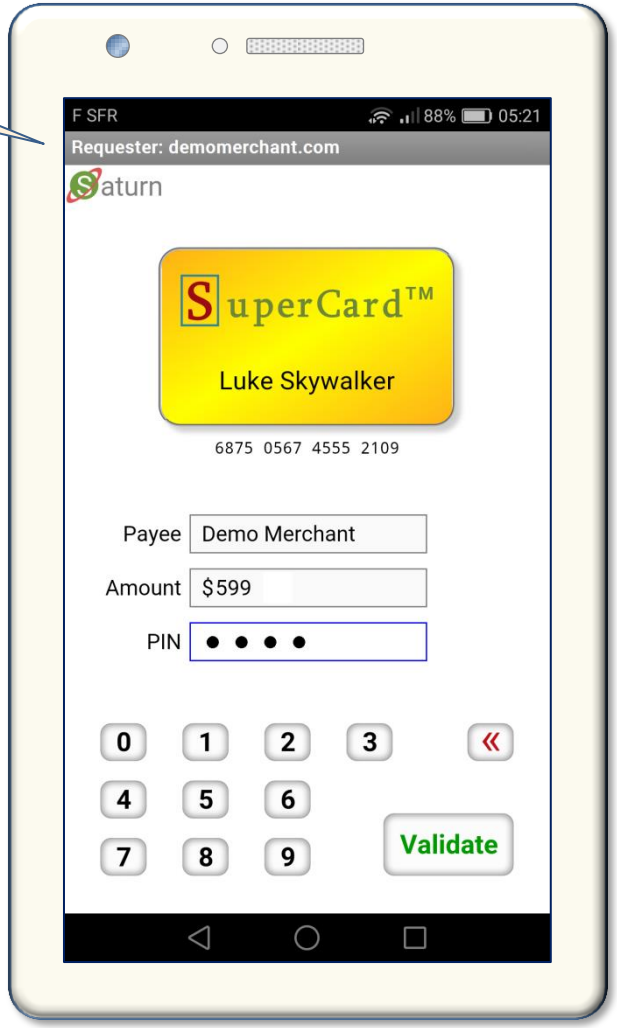


Processes

Keys

Typical Applications

Secure payments on the web as well as in brick and mortar shops



e-Government

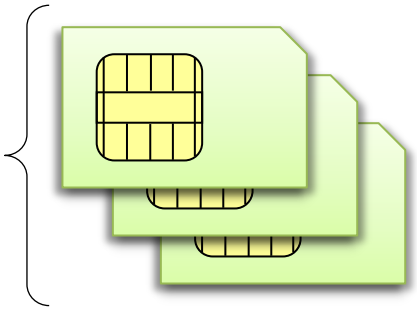
Income declaration

Year: 2016
Name: Marion Anderson
Citizen code: 19950710-1518
Declared income: Fair ($30000-$99999)

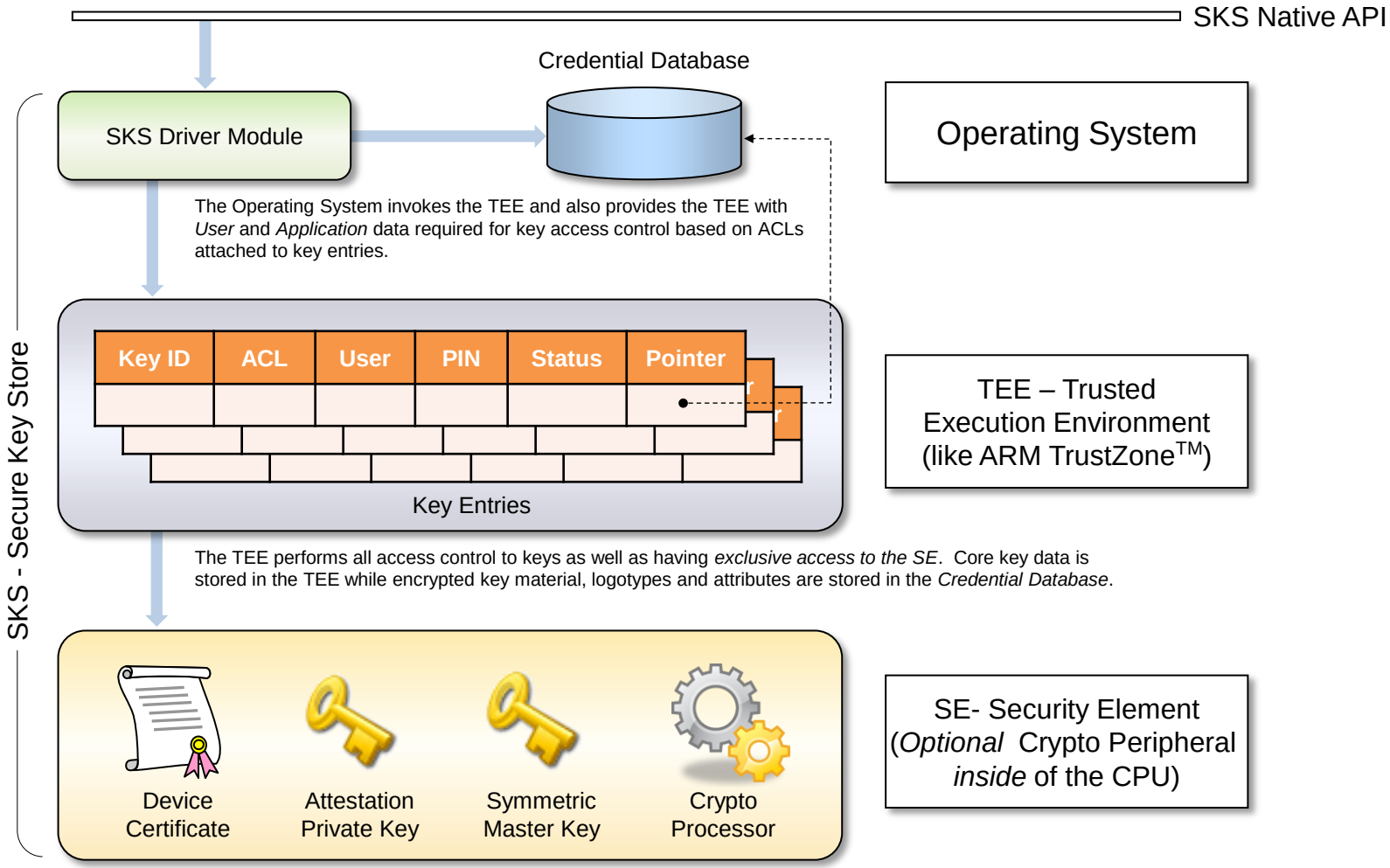
eGovernment signature applications like *income declarations*, *change of address* and *permits requests*

Virtual SIM-cards enable you to buy, carry and use *multiple subscriptions* in an easy way

Virtualized SKS SIM Credentials =



Secure Key Store – A Three Layer Architecture



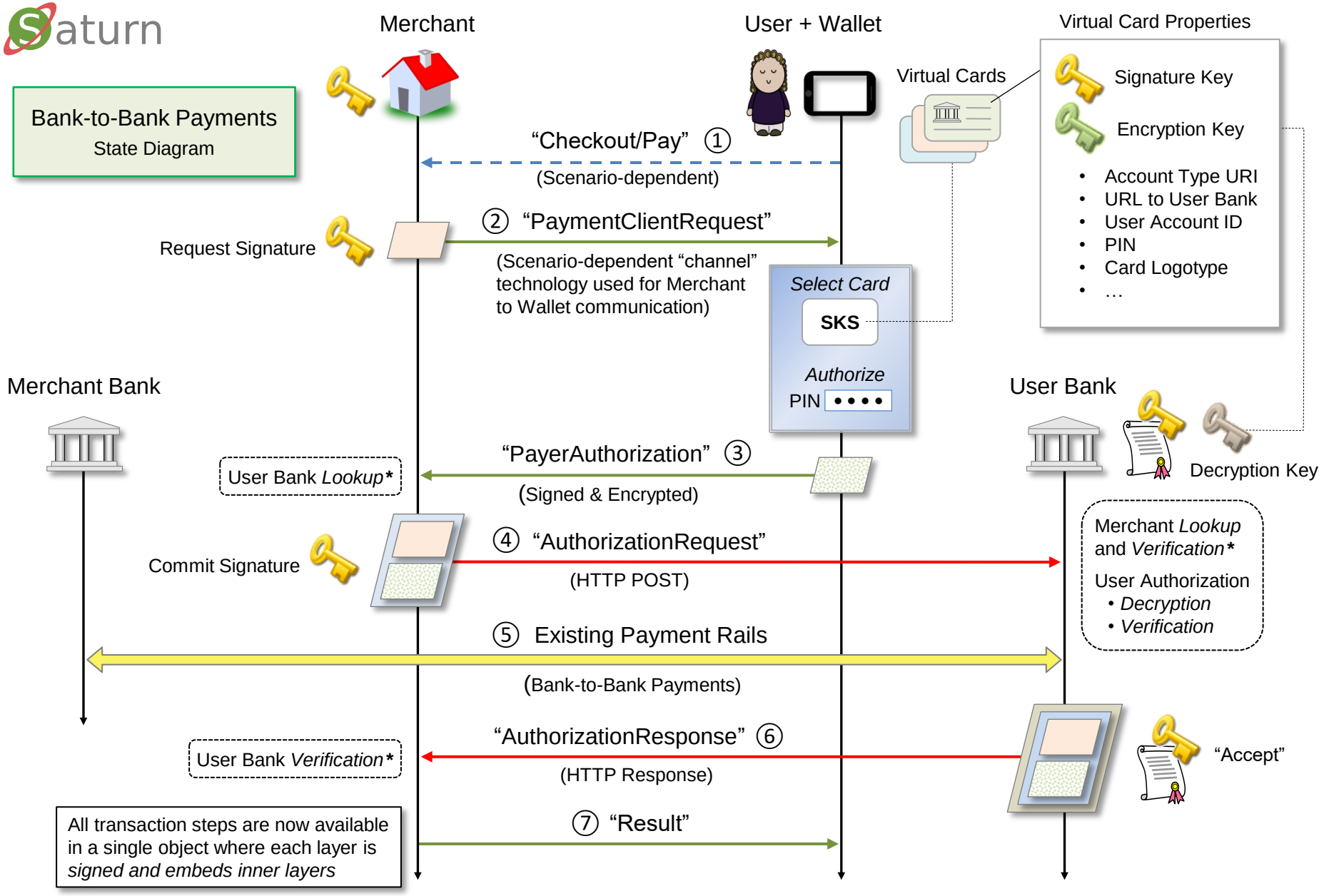
The SE only holds static data: a *Device Certificate,* a matching *Attestation Key* and a *Master Key* which is used for wrapping user keys. The *Attestation Key* signs session keys which KeyGen2 uses for secure key provisioning and management. The SE generates wrapped keys and as well as performing standard cryptographic operations on wrapped keys. That is, *user keys are never exposed in clear*.

Key + “Decoration” = Credential

Element	Description
	*Mandatory*: Asymmetric (private) or Symmetric (secret) key
	*Mandatory*: X.509 certificate having two uses: • Support for PKI-based applications • Providing a “name” for key management operations
Algorithms	*Optional*: Set of algorithms permitted to use with the key
Images	*Optional*: For usage in GUIs. Type information enable selecting appropriate images for different scenarios 
PIN	*Optional*: For key unlock. May be substituted or complemented with biometrics if the hardware supports that
Attributes	*Optional*: Arbitrary text and binary properties containing things like *URIs*, *Public keys*, and *Constants* to be used by associated applications
ACL	*Optional*: Access Control List protecting keys from illicit access

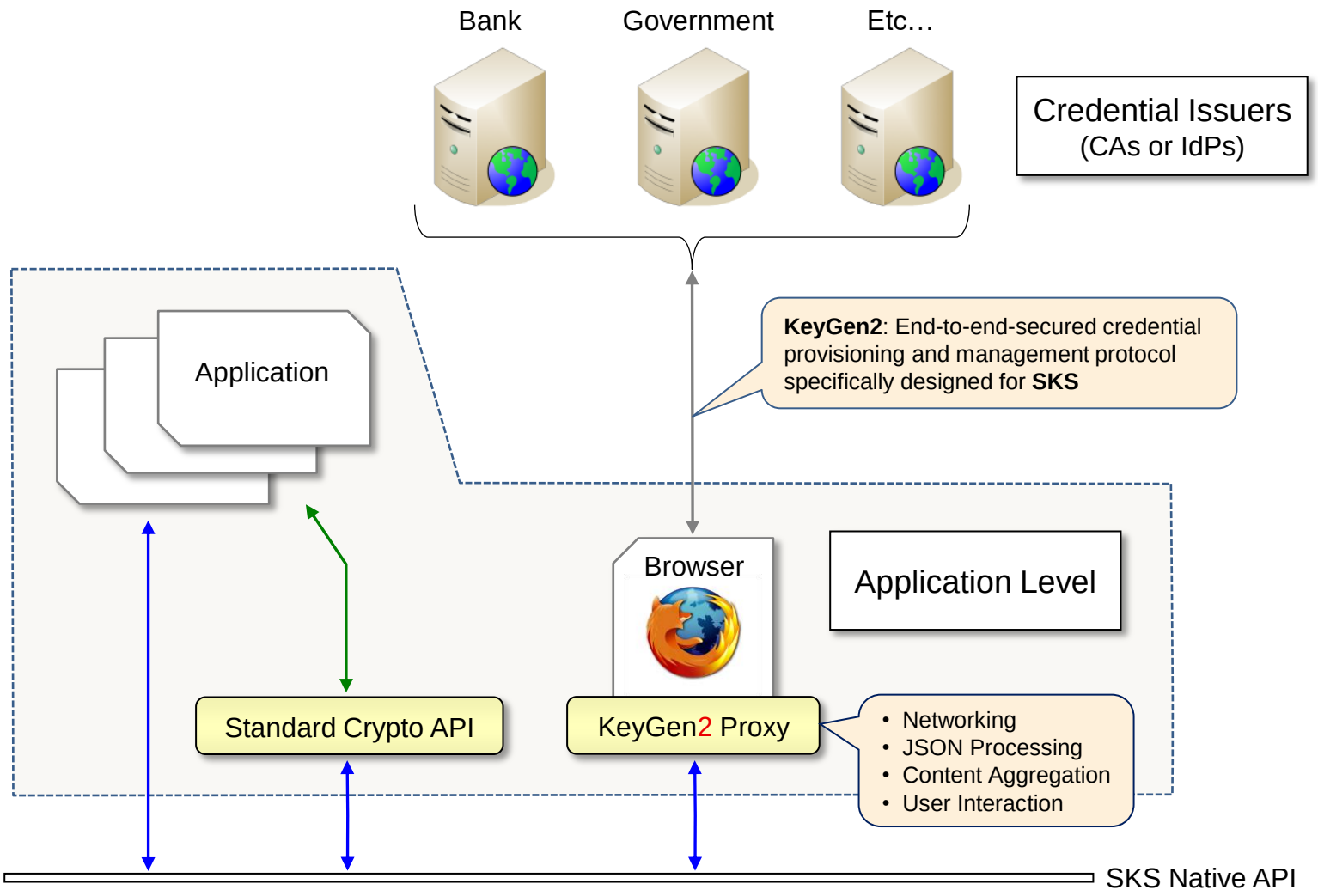
Code	*Not Supported.* Trusted Credentials != Trusted Applications

Demo - Saturn (Payment Authorization)



Sample application that was built using **SKS** and **KeyGen2** for *Storing/Using* respectively *Issuing* Virtual Cards

The Missing Link – Credential Provisioning



Demo – Enrollment using KeyGen2

Project Status – February 2017

- SKS software emulator in Java
- Android “App” implementing SKS, KeyGen2 and two test applications available on PlayStore
- Public test applications on the Web
- Extensive documentation
- Published on GitHub: https://github.com/cyberphone

Currently Missing

- SKS/TEE integration
- Browser integration
- *and most of all, device vendor partners…*

Related Standardization Efforts

JCS – JSON Clear-text Signature. Fully implemented reference implementation in Java. JCS also runs in *browsers* and *Node.js*


```
{
  "myProperty": "Some data",
  "signature": {
    "algorithm": "ES256",
    "publicKey": {
      "type": "EC",
      "curve": "P-256",
      "x": "vlYxD4dtFJOp1_8_QUcieWCW-4KrLMmFL2rpkY1bQDs",
      "y": "fxEF70yJenP3SPHM9hv-EnvhG6nXr3_S-fDqoj-F6yM"
    },
    "value": "gNfr9Es0cnc263tmOYMsctBh … Qd2h8QSePPGsKdkLILVJDBlAbkQ1eA"
  }
}
```