

# On-board Credentials with open provisioning

**N. Asokan**

**Nokia Research Center, Helsinki**

Joint work with Jan-Erik Ekberg, Kari Kostianen, Pekka Laitinen, Arne Rantala (VTT)

**NOKIA**

# Outline

- On-board Credentials (ObCs): What and Why
- ObC Architecture
- Secure Provisioning of ObCs
- Instantiations of the Architecture
- Deployment Considerations
- ObCs in Action
- Status

This is a talk about a research project. Opinions stated here do not necessarily imply Nokia's official strategy

# On-board Credentials: What and Why

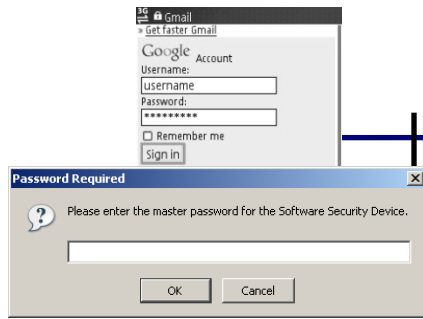
# On-board Credentials (ObCs)

*open*  
An credential platform that leverages on-board trusted execution environments

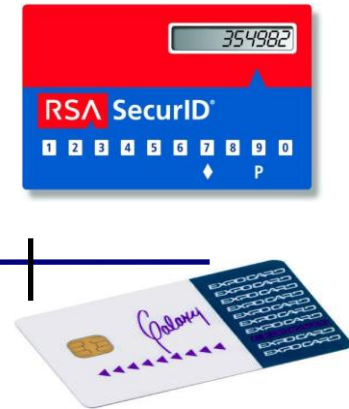


*Secure yet inexpensive*

# On-board Credentials



SW-only credentials



HW credentials

Similar in spirit to multi-application smartcards

- But **without any issuer control**

Design should take limitations of existing trusted hardware into account

# Design goals

*Credential = program + secret*

- Credential programs can be **executed securely**
  - Use a trusted execution environment (TrEE)
- Credential **secrets can be stored securely**
  - Use a device-specific secret in TrEE for secure storage
- Creating **new types of credentials** (i.e., new credential programs) is easy
- **Anyone can create and use** new credential types
  - Need a security model to strongly isolate credential programs from one another
  - Avoid the need for centralized certification of credential programs
- **Anyone can provision credential secrets** securely to a credential program
  - Need a mechanism to create a secure channel to the credential program
  - (certified) device keypair; unique identification for credential programs

# ObC Architecture

# ObC architecture

*Credential = program + secret*

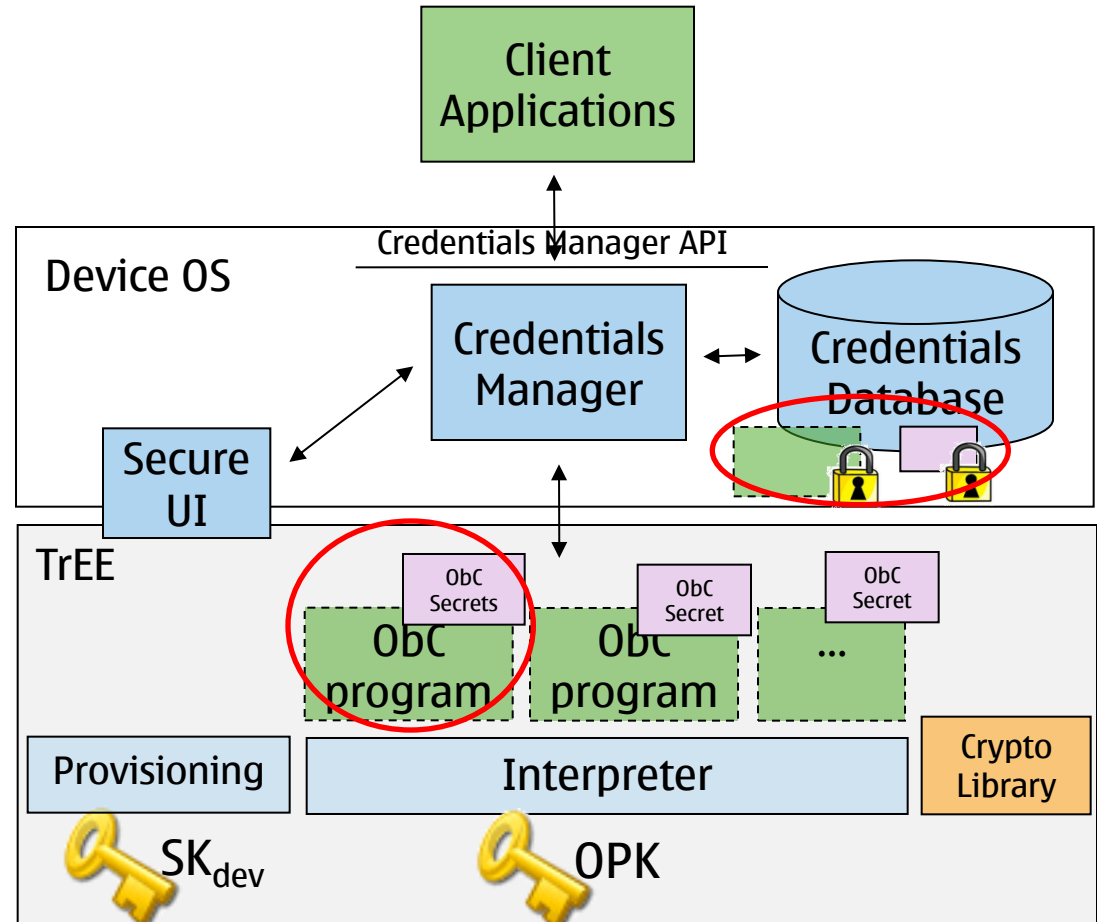
TrEE based on trusted hardware

- **Secure storage**, using a device-specific ObC Platform Key (OPK)
- **secure execution**
- **Certified device keypair**  
 $PK_{dev}/SK_{dev}$

**Interpreter** runs in TrEE

- Credential programs (ObC programs) are **scripts**
- program hash as unique ID

Encrypted storage for credentials  
(secrets + programs)



See [ACM ASIACCS '09 paper](#) (or the longer [technical report](#))

# Isolation of ObC Programs

Isolating the platform from programs

- Constraining the program counter, duration of execution, ...

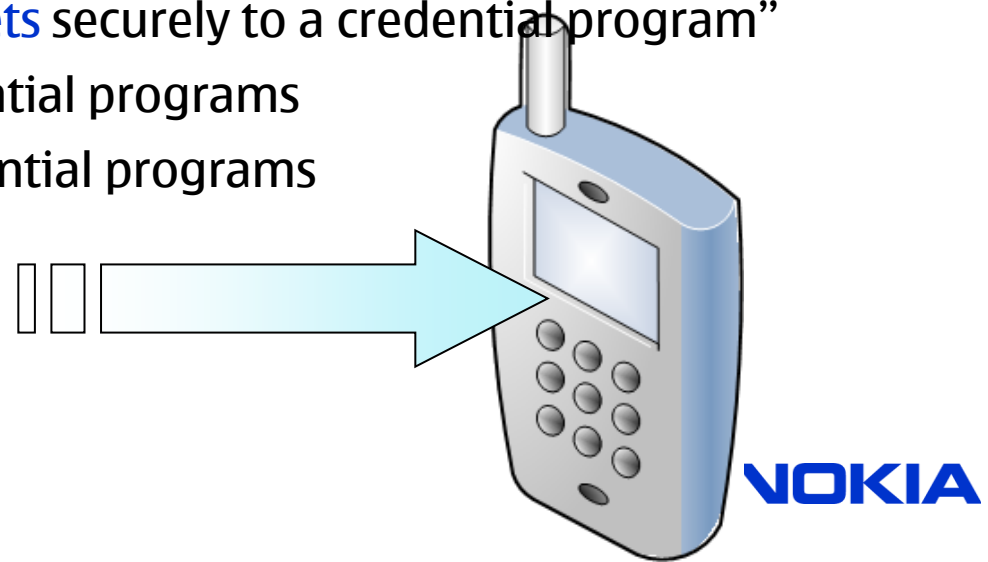
Isolating programs from one another

- Only one ObC program can execute at a time
- An ObC program can “seal” data for itself
  - Sealing key is different for every independent ObC program  
Sealing-key = KDF (OPK, program-hash)
  - A program can invoke functions like “seal(data)” (unsealing happens automatically on program loading)

# Secure Provisioning of ObCs

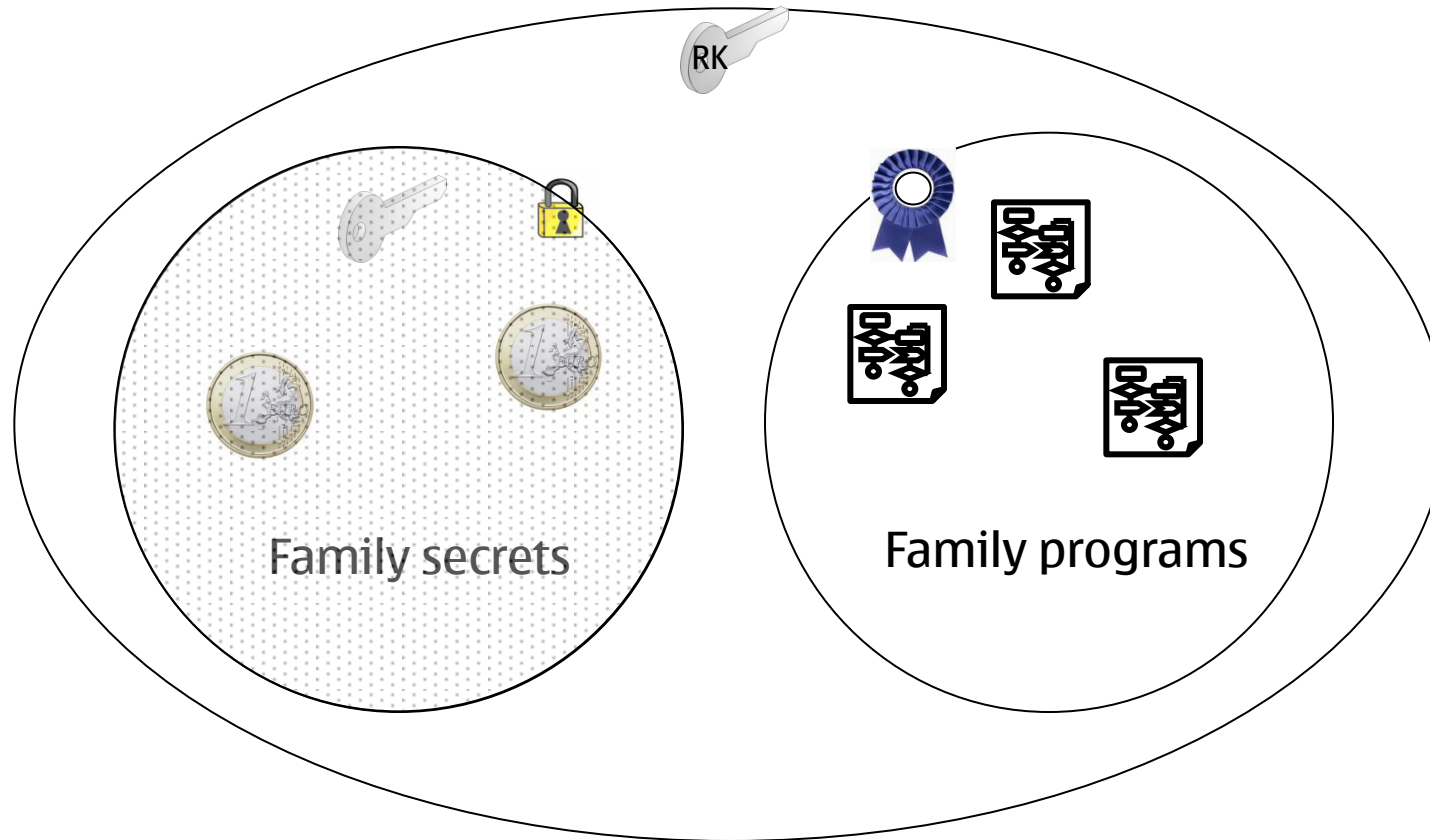
# Requirements for Provisioning Credential Secrets

- Provisioning protocols typically focus on **user authentication** only
  - CT-KIP, Open Mobile Alliance Device Management (OMA DM), ...
- IETF keyprov working group is defining Dynamic Symmetric Key Provisioning Protocol (DSKPP)
  - Allows **device authentication** as well
- We need more...
  - provision a key so that it can be accessed by **specific credential programs**
- Subject to...
  - “**Anyone can provision credential secrets** securely to a credential program”
  - Support for multiple versions of credential programs
  - Support for several co-operating credential programs



# Provisioning credential secrets (1/4)

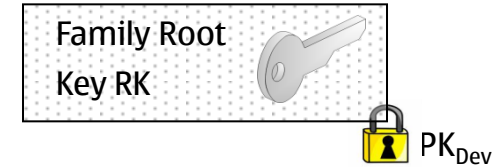
Basic Idea: the notion of a **family** of credential secrets and credential programs endorsed to use them



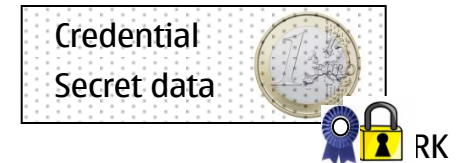
# Provisioning credential secrets (2/4)

- Provision a family **root key** to the device
  - using **authentic device public key**  $PK_{Dev}$
- Transfer encrypted credential secrets
  - using authenticated encryption (AES-EAX) with RK
- Endorse credential programs for family membership
  - Program ID is a cryptographic hash of program text
  - using authenticated encryption (AES-EAX) with RK

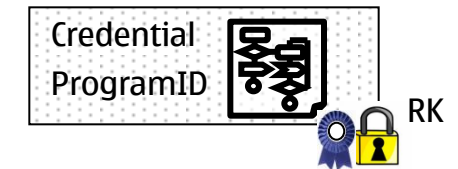
ObCP/Init



ObCP/Xfer



ObCP/Endorse



# Provisioning credential secrets (3/4)

- Anyone can define a family by provisioning a root key
- Multiple credential secrets and programs can be added to a family
- Credential Programs can be encrypted as well



Tech Report at <http://research.nokia.com/files/NRCTR2008007.pdf>

# Provisioning credential secrets (4/4)

Verifies  $Cert_D$ , Creates new RK

$Init = Enc(PK_D, RK)$

$Xfer = AE(RK, secret)$

$Endorsement = AE(RK, hash(program))$

$LFK = KDF(RK, OPK)$

$LEK = KDF(OPK, hash(program))$

$ET = AE(LEK, LFK)$

Credential issuer

$PK_D, Cert_D$

Init, Xfer, Endorsement

$PK_D/SK_D =$  device keypair

$Cert_D =$  manufacturer certificate for  $PK_D$

$RK =$  family root key

$LFK =$  local family key

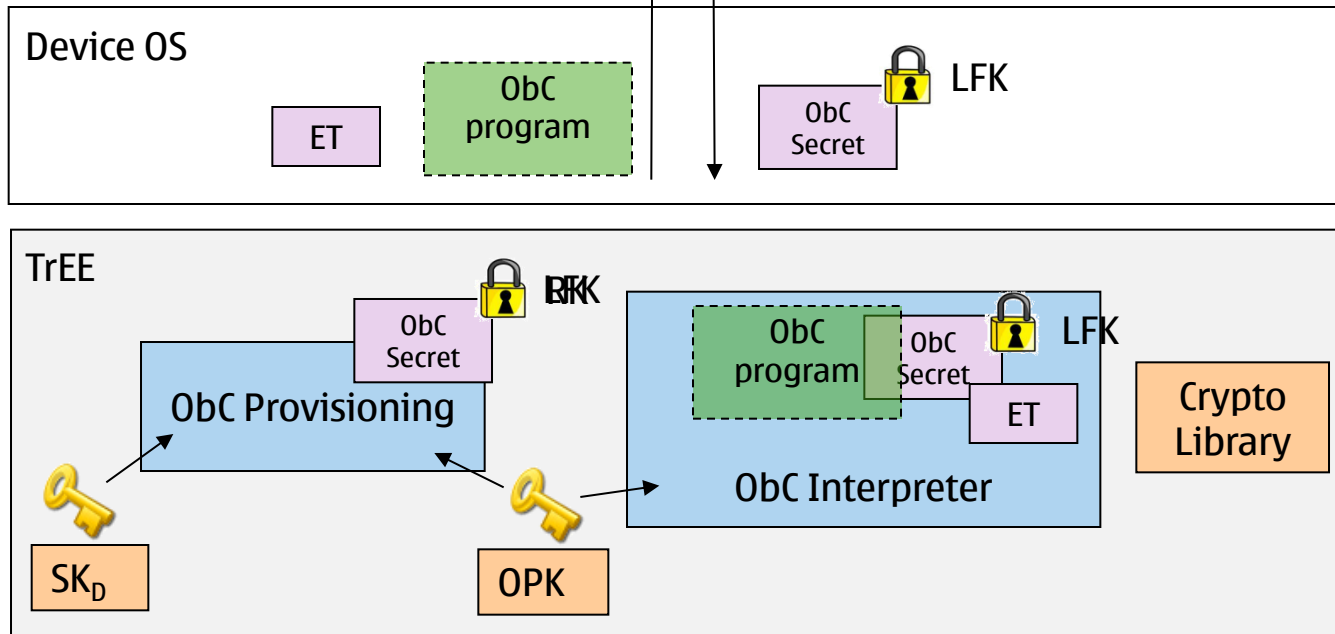
$LEK =$  local endorsement key

$ET =$  endorsement token

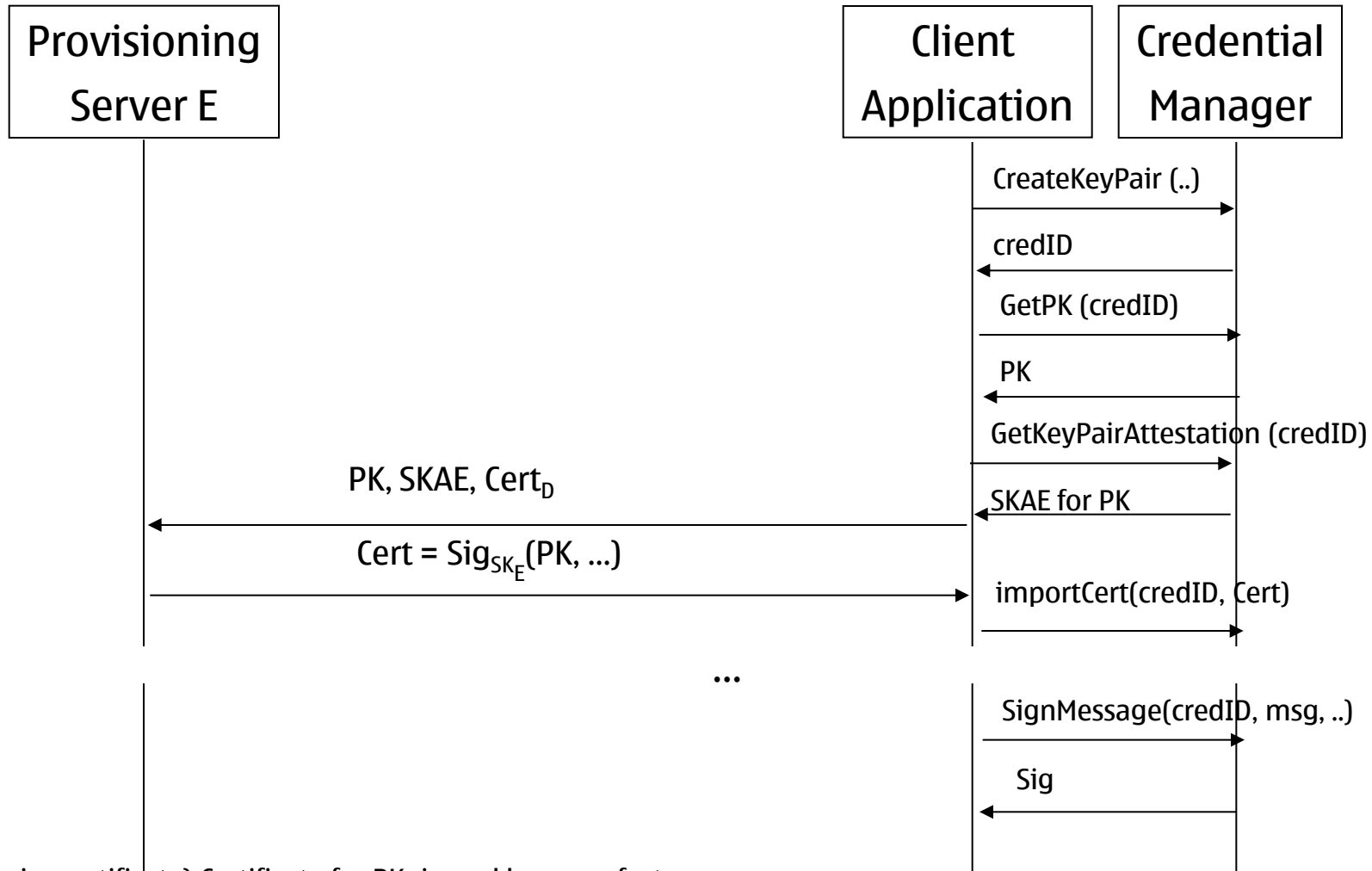
$Enc =$  public key encryption

$AE =$  authenticated encryption

$KDF =$  key derivation function



# Asymmetric ObCs



Cert<sub>D</sub> (Device certificate) Certificate for PK<sub>D</sub> issued by manufacturer

SKAE (Subject Key Attestation Evidence) for PK: Signature on PK issued by SK<sub>D</sub>, attesting that SK is within the TrEE

# Instantiations of the Architecture

# M-Shield™: Example hardware TrEE #1

M-Shield provides

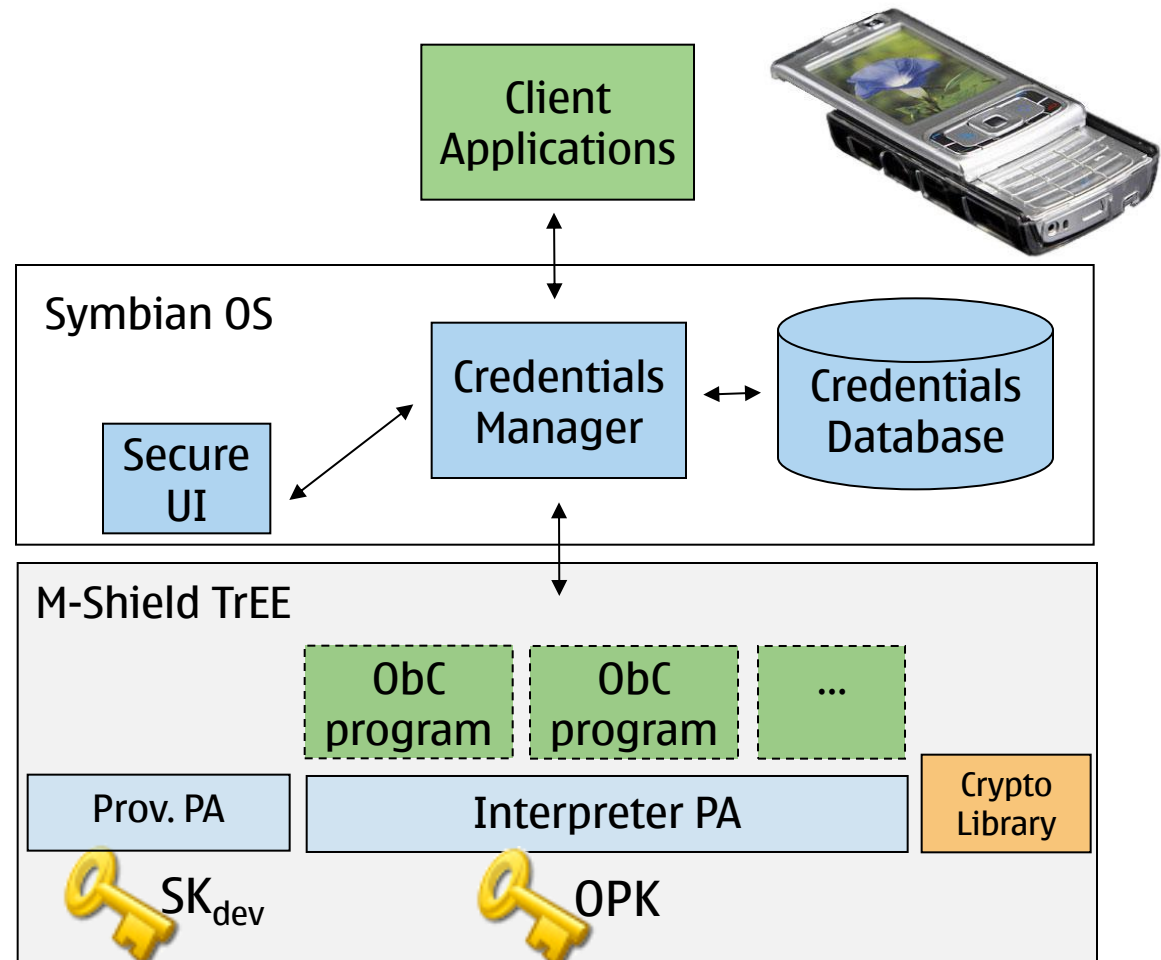
- Secure boot
- Chip-specific secret key (e-fuse)
- Secure execution of certified “Protected Applications” (PAs)
- On-chip RAM for PAs
- ... (hardware RNG, crypto accelerators, ...)



[http://focus.ti.com/pdfs/wtbu/ti\\_mshield\\_whitepaper.pdf](http://focus.ti.com/pdfs/wtbu/ti_mshield_whitepaper.pdf)

# ObC on Symbian/M-Shield secure h/w (2007-2009)

- M-Shield secure boot used for validation of OS (Symbian OS)
- Interpreter (Lua), Provisioning subsystem are PAs
  - Use on-chip RAM
- OPK from chip-specific secret
- Device key pair
  - generated by Prov. PA
  - protected by chip-specific secret key
  - certified by manufacturer



# TPM: Example hardware TrEE #2

TPM provides

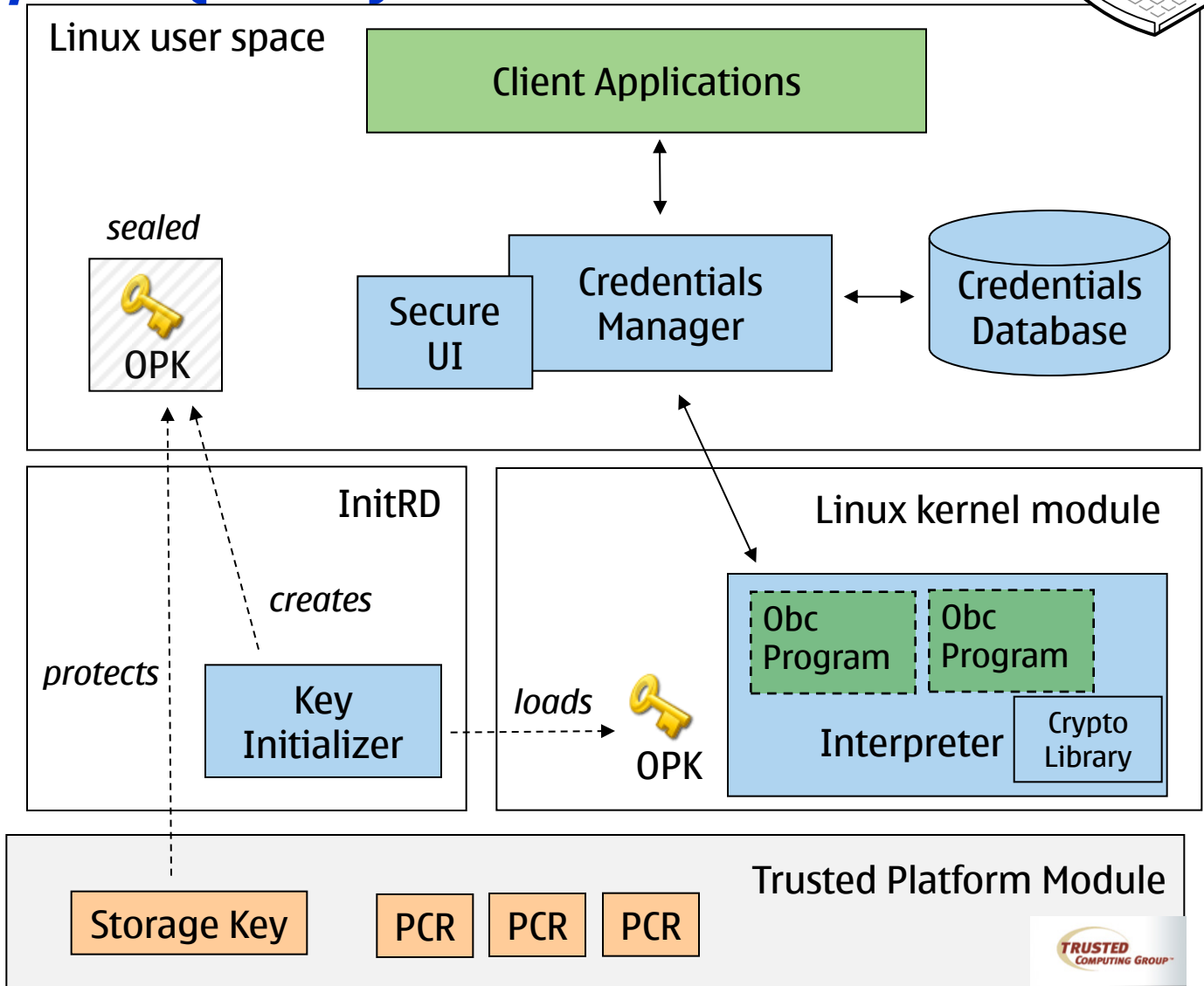
- Authenticated boot
  - Components during boot measured and recorded in Registers (PCRs) within TPM
  - A set of PCR values = a “configuration”
- Secure storage for keys bound to a specific configuration
- Ability to seal arbitrary data bound to a specific configuration
- Secure execution of selected cryptographic operations
- ... (remote attestation, ...)



# ObC using Linux/TPM (2006)



- Interpreter in kernel module on InitRD
- KeyInitializer in InitRD creates OPK on first use and seals for current configuration
- KeyInitializer unseals OPK on subsequent invocations.
- Security of execution improved using dynamic root of trust (2009)



MSc thesis work:

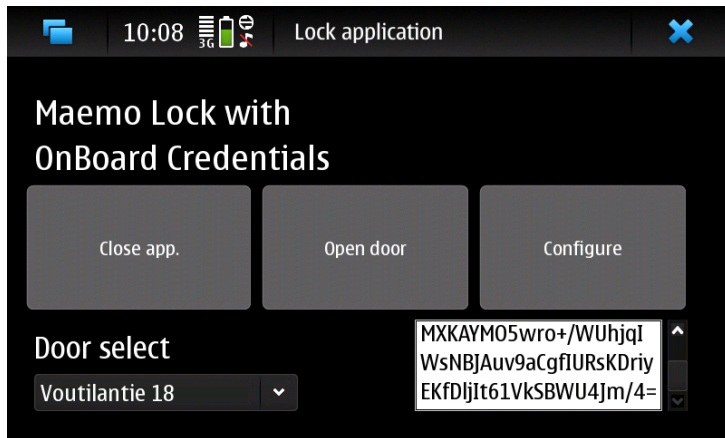
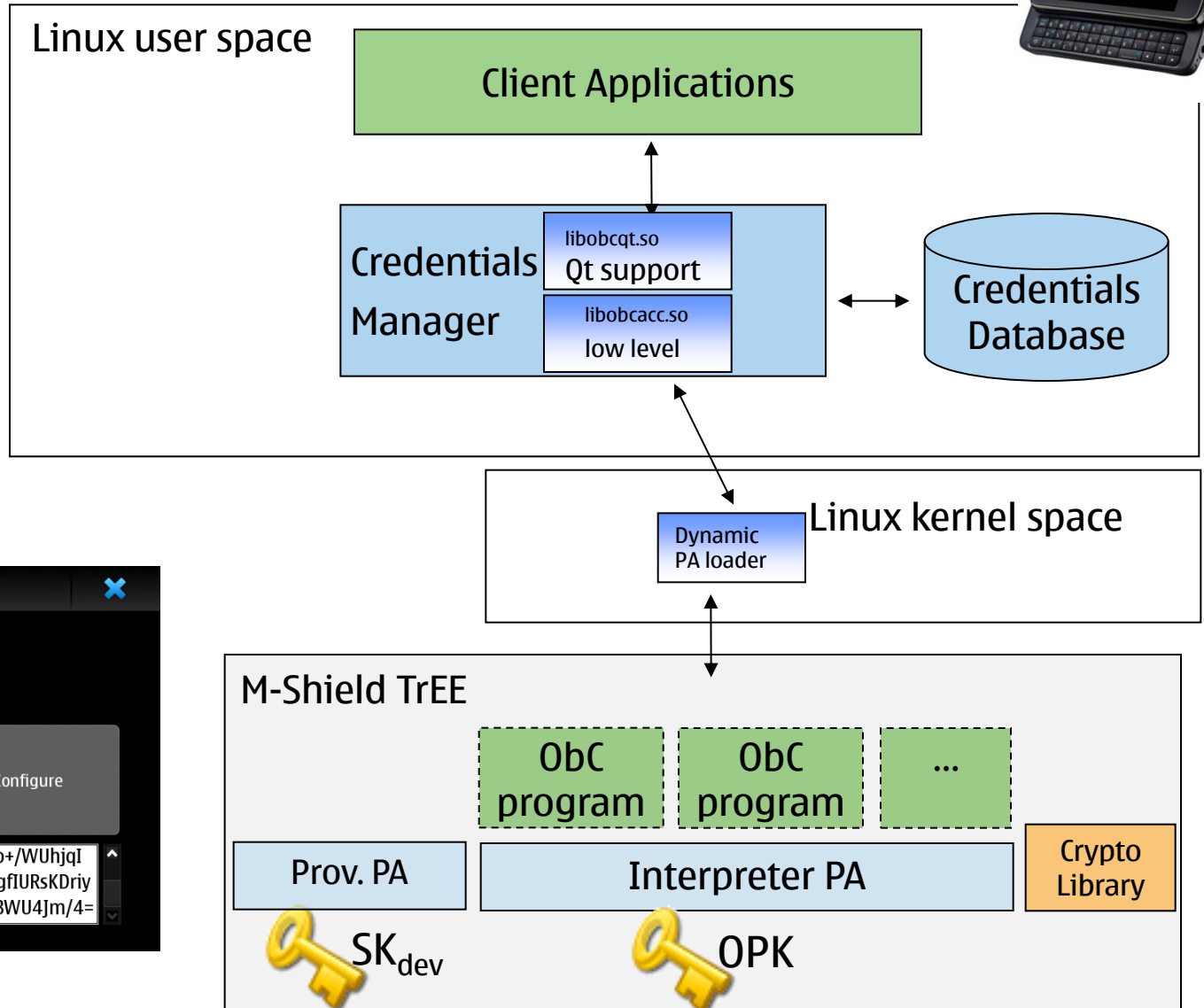
<http://asokan.org/asokan/research/Aish-Thesis-final.pdf>



# ObC on Maemo/M-Shield secure h/w (2009-2010)



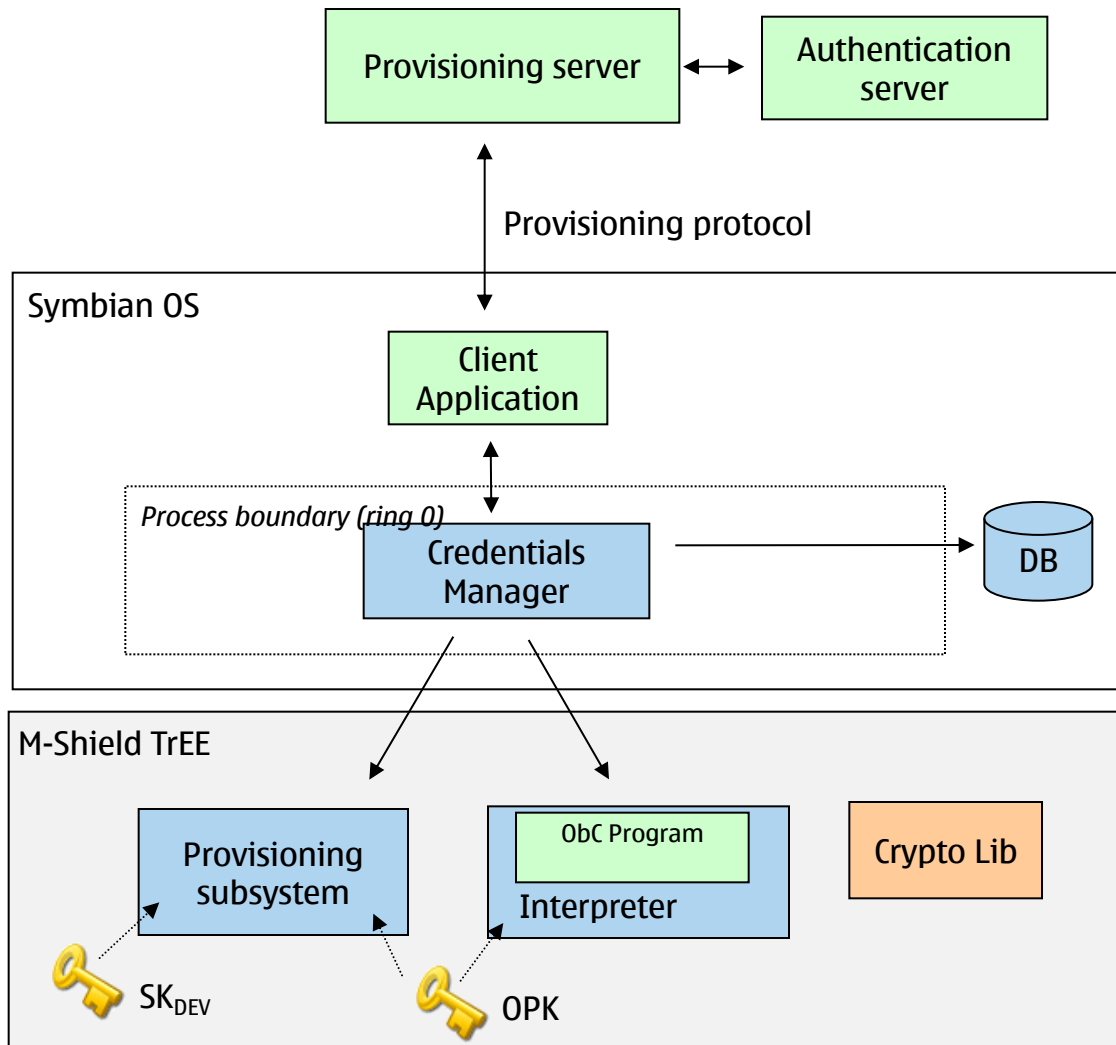
- Maemo port in progress
- Credentials Manager interface as a Qt API



# Deployment considerations

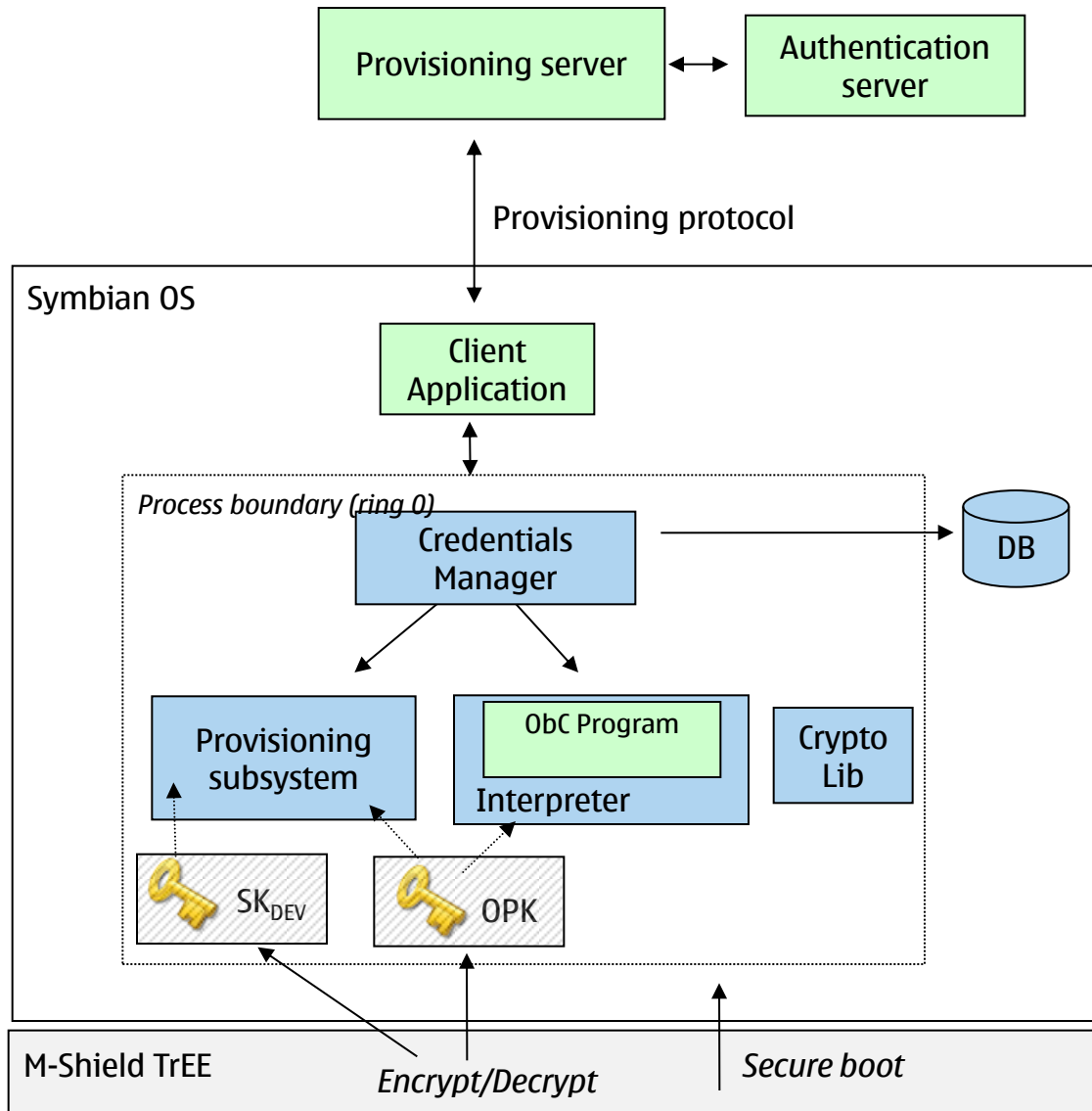
Skip to [“ObCs in action”](#)

# 1. ObC: Full use of secure HW



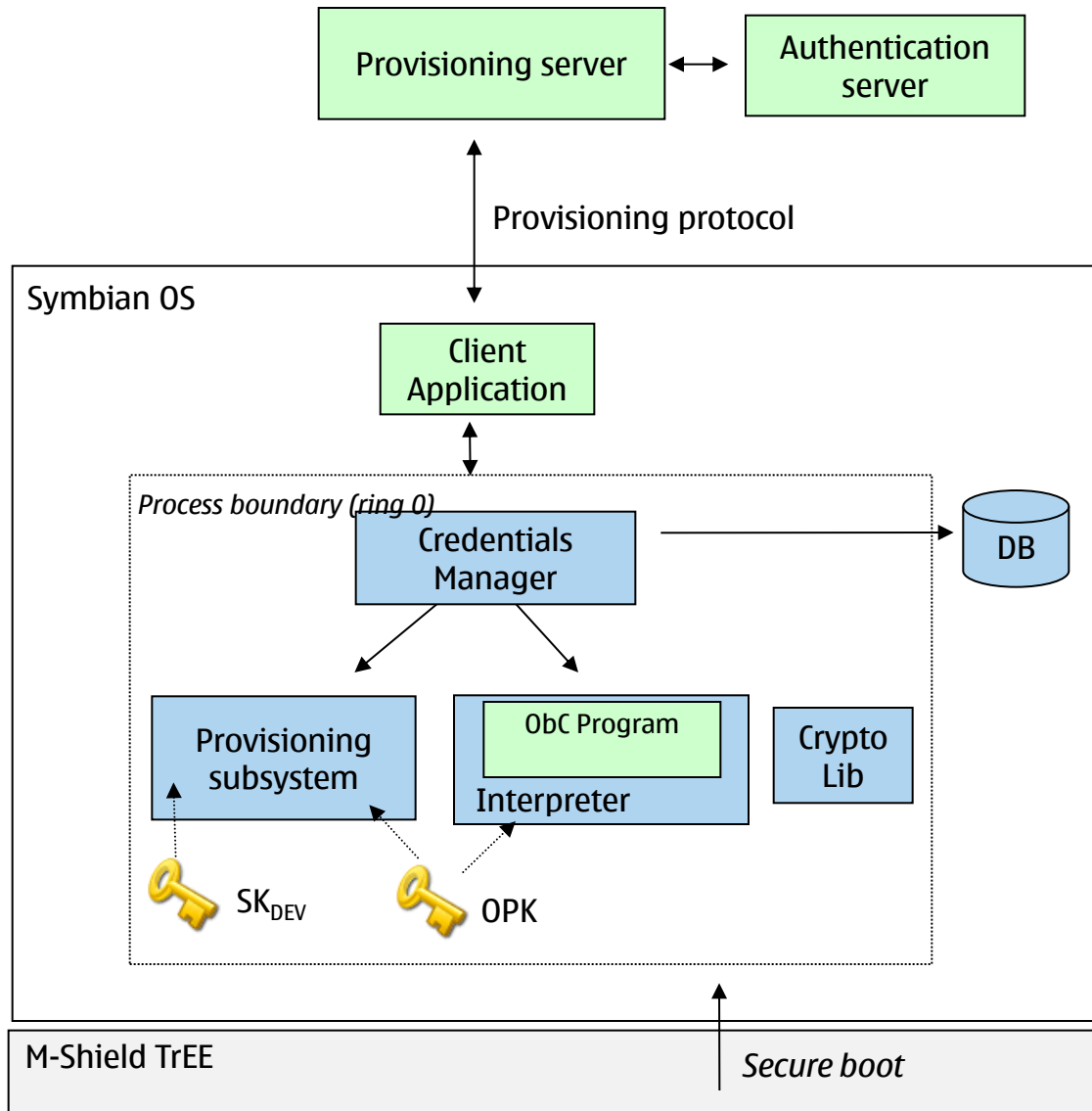
- ObC secret and algorithm (ObC program) protected by hardware TrEE
  - Provisioned data encrypted with  $PK_{Dev}$
  - Not accessible to Device OS
  - Cannot be copied between devices
  - Hardware attack is typically destructive and device-specific
- Encrypted secret stored in Credentials Manager database
  - Can be backed up
- Currently works on devices with R&D certificates (tested on N95/N96)

## 2. ObC: Partial use of secure HW



- Credential Manager uses secure HW for encrypting its device and storage key
  - Credentials Manager can do this if it has Nokia Vendor ID (VID)
- ObC secret stored in Credentials Manager database protected by use of hardware TrEE
- ObC program runtime execution protected by Symbian OS platform security
- Works on most recent off-the-shelf Symbian devices

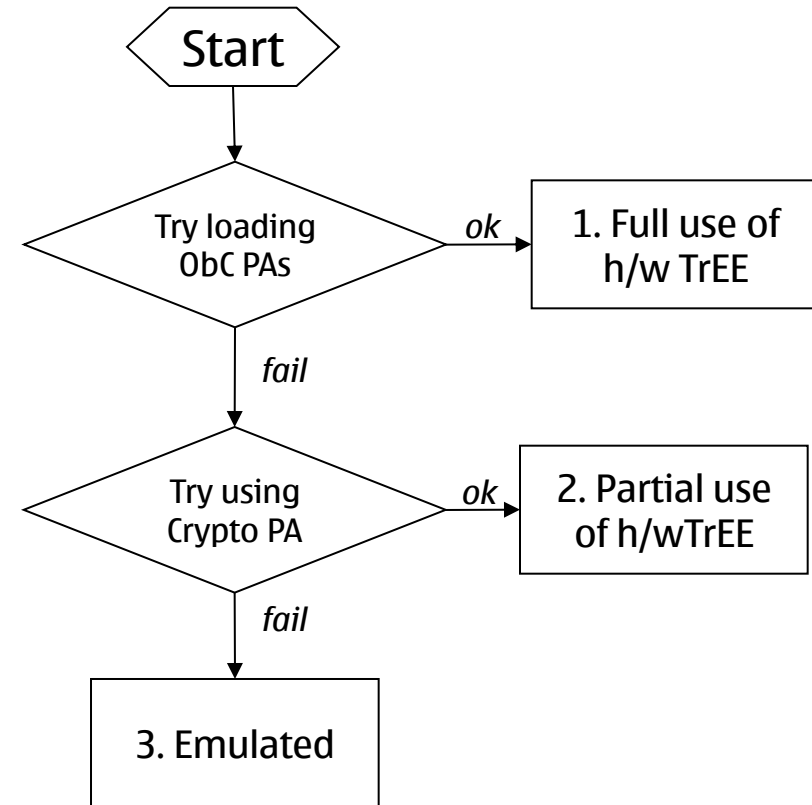
# 3. ObC: Emulated



- The Credentials Manager process contains also Interpreter and Provisioning subsystem
- ObC program executed in Symbian OS
- ObC secret and program protected by Symbian platform security
  - Hardware TrEE for secure boot
- Works on any S60 3.1+ phone

# ObC implementation supports all 3 variants

- Implementation contains code for emulating TrEE PAs (interpreter+provisioning+crypto) in S60
- Same software package can be installed in any Symbian/S60 device
  - automatically decides the variant to use
- (“PA” = “Protected Application” refers to code that runs in the M-Shield hardware TrEE)



# ObCs in action

# An Example ObC: SecurID one-time password authentication



*Joint research project with RSA security*

# Phone as smartcard (PASC)

- Applications use public key (PK) cryptography via standard frameworks
    - Crypto API (windows), Cryptoki (Linux, Mac), Unified Key/cert store (Symbian)
    - Agnostic to specific security tokens or how to communicate with them
- Any PK-enabled smartcard can be used seamlessly with PK-aware applications!



What if mobile phone can present itself as a PK-enabled smart card?

# Demo: Mobile device as secure token for VPN access

  
Väestörekisterikeskus  
Befolkningsregistercentralen  
“Kansalaisvarmenne”  
PKCS#15 – smart card



  
FUJITSU  
mPollux

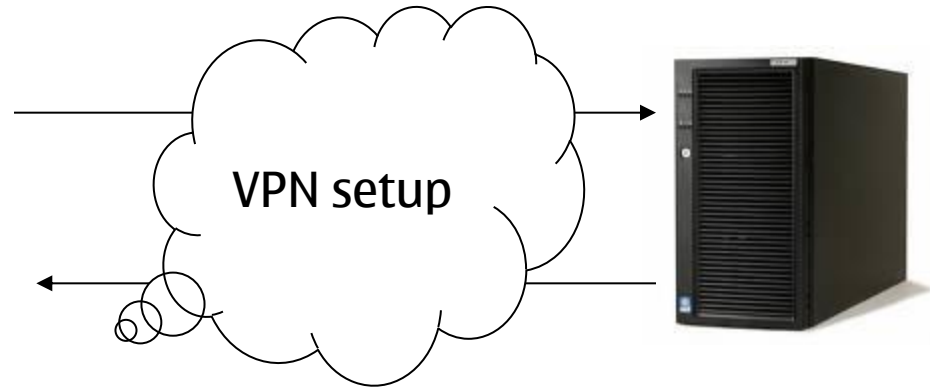


Sign

signature



Secure yet inexpensive



# ObC Status

# ObC Status (1/2)

ObC available for Symbian devices

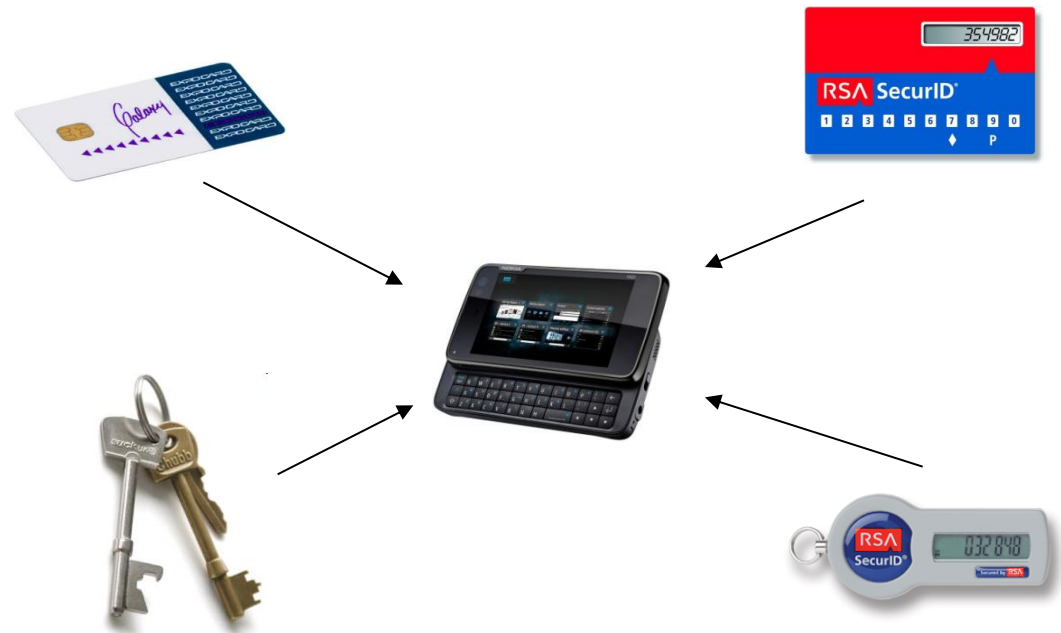
- SIS package for any recent (3.2 or 5.0) device (uses hardware TrEE in N95/96)
- Development environment for ObC programs (Windows)
- Credential Manager and interfaces (native, python, javascript)
- Available under limited license agreement for research and testing

# ObC Status (2/2)

- Maemo implementation demonstrated in December 2009
  - Credential Manager Qt interface
- Possible low-end implementation in 2010
- On-going research to address several extensions
  - E.g., Credential transfer, validation, new types of hardware TrEEs
- Useful for several applications
  - Device authentication, financial services, secure messaging, ...
  - Pragmatic means to solve otherwise hard privacy/security problems in distributed computing (e.g., secure multi-party computation)

# Summary

- On-board Credentials platform
  - inexpensive
  - open
  - secure
- A step towards the vision of a **personal trusted device**
- Available for you to build on



How to make it possible to build trustworthy information protection mechanisms that are simultaneously **easy-to-use** and **inexpensive** to deploy while still guaranteeing **sufficient protection**?

