On-board Credentials

N. Asokan
Nokia Research Center, Helsinki

Joint work with Jan-Erik Ekberg, Kari Kostiainen, Pekka Laitinen, Aarne Rantala (VTT)
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)

A credential platform that leverages on-board trusted execution environments.

Secure yet inexpensive
On-board user credentials: what and why

On-board Credentials

SW-only credentials

Dedicated HW credentials
On-board user credentials: design goals

- 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
- 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
- Protection of asymmetric credentials is attestable to anyone
  - Anyone can verify that a private key is protected by the TrEE

Credential = program + secret
ObC Architecture
ObC Architecture

On Trusted Execution Environments (TrEEs) with

- Secure execution (within TrEE)
- Secure storage (secret key in TrEE)
- Certified device keypair (PK_{dev}/Sk_{dev} in TrEE)
On Trusted Execution Environments (TrEEs) with

- Secure execution (within TrEE)
- Secure storage (secret key in TrEE)
- Certified device keypair ($PK_{dev}/SK_{dev}$ in TrEE)

More in ACM ASIACCS ‘09 paper
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
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
Provisioning credential secrets (4/4)

Verifies $\text{Cert}_D$, Creates new $\text{RK}$

$\text{Init} = \text{Enc}(\text{PK}_D, \text{RK})$

$\text{Xfer} = \text{AE}(\text{RK}, \text{secret})$

$\text{Endorsement} = \text{AE}(\text{RK}, \text{hash(program)})$

$L\text{FK} = \text{KDF}($$\text{RK}, \text{OPK})$

$L\text{EK} = \text{KDF}(\text{OPK}, \text{hash(program)})$

$\text{ET} = \text{AE}(\text{LEK}, \text{LFK})$

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

© 2007-2010 Nokia, ObC-overview-public-jul2010 N. Asokan
Asymmetric ObCs

Provisioning
Server E

Client
Application

Credential
Manager

PK, SKAE, Cert_D

Cert = Sig_{SK_E}(PK, ...)

Cert_D (Device certificate) Certificate for PK_D issued by manufacturer
SKAE (Subject Key Attestation Evidence) for PK: Signature on PK issued by SK_D, 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, ...)

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

- M-Shield secure boot used for validation of OS
- Interpreter, 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, ...)

© 2007-2010 Nokia, ObC-overview-public-jul2010 N. Asokan
ObC using Linux/TPM (2006, 2009)

- 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.

MSc thesis work:
ObC on Maemo/M-Shield secure h/w (2009-2010)

Linux user space

- Credentials Manager API
  - Qt API (libDeviceEngine.so, libKeyPairEngine.so)
  - Low level C API (libobcacc.so)

ObC Daemon (obcsrv)

App specific ObC Database

Device specific ObC Database

Linux kernel space

- BBS Security Driver

M-Shield TrEE

- NOPPA PA
- ObC program
- Interperter PA
- Crypto Library

- Prov. PA, RSA PA,
- SKdev
- OPK

Maemo Lock with OnBoard Credentials

© 2007-2010 Nokia, ObC-overview-public-jul2010 N. Asokan
Deployment considerations
1. **ObC: Full use of secure hardware**

- **ObC secret and algorithm (ObC program) protected by hw TrEE**
  - $PK_{Dev}$ to protect provisioning or attestation
  - Secrets not accessible to OS
  - Cannot be copied between devices
  - Hardware attack typically destructive and device-specific

- **Encrypted secret stored in Credentials Manager database**
  - Can be backed up

- **Works on certain existing device models (e.g., N900)**
2. ObC: Partial use of secure hardware

- ObC PAs emulated in the Credential Manager (OS process)
- Secure HW used to enable secure storage and device authentication
- ObC program runtime execution protected by OS platform security
- Works on most recent off-the-shelf Symbian devices
3. ObC: Emulated

- ObC PAs emulated in the Credential Manager (OS process)
- Secure HW may be used for secure boot
- Storage ObC secrets and ObC program runtime execution protected by OS platform security
- No device authentication
- For debugging/development
ObC implementation supports all 3 variants

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

Secure yet inexpensive

TLS client auth

Sign

signature

“Kansalaisvarmenne”
PKCS#15 – smart card

“Kansalaisvarmenne”
PKCS#15 – smart card

Secure yet inexpensive

© 2007-2010 Nokia, ObC-overview-public-jul2010 N. Asokan
Demo: Remote Attestation

Challenge, attestation data, sig(chal., att. data)

Mobile terminal proves that it is running the right widget in the right web runtime.

ObC

Run attestation

Getting challenge from server...

Challenge: 0B52hH2pMwzD04eWGMJg==
SID: 10282322
WID: 1019657
Widget ID: com.nokia.research.trumps.AttestationDemo

Attestation:
mj8gucGjYjYuqHjqJk3gDULJ36jUR3dPwE6xGTvJc25Jf31V6ukC
Hj0WpaePPDAvxBeum/XMaUqi+D6HkctshrD0V5eejJbCgCm0q3aOWC
mZ8FrHfyUZAOWhHgf/hCVYngfc=

Sending attestation data to server for validation...

Attestation result from server: SUCCESS

Options Exit
ObC Status
ObC Status (1/2)

ObC available for Symbian and Maemo

- Binary package installable on any recent Nokia Symbian/Maemo device
- Development environment for ObC programs (Windows, Linux)
- Credential Manager and interfaces (native, python, javascript)
- Available under limited license agreement for research and testing
ObC Status (2/2)

- 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?