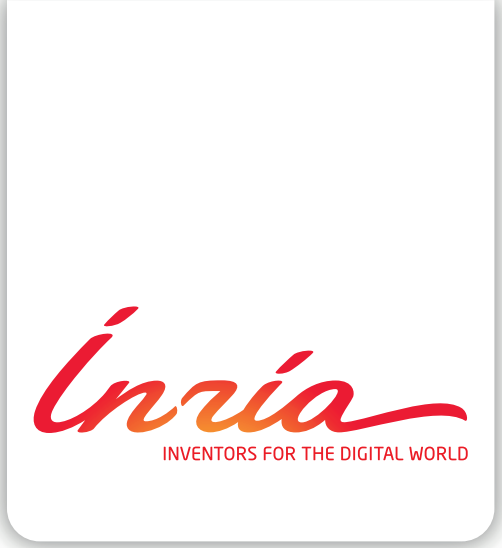


Analysing Cryptographic Hardware Interfaces with Tookan

Graham Steel

joint work with R. Bardou, M. Bortolozzo, M. Centenaro, R. Focardi, Y. Kawamoto, L. Simionato, J.-K. Tsay

Analysing Security Device Interfaces

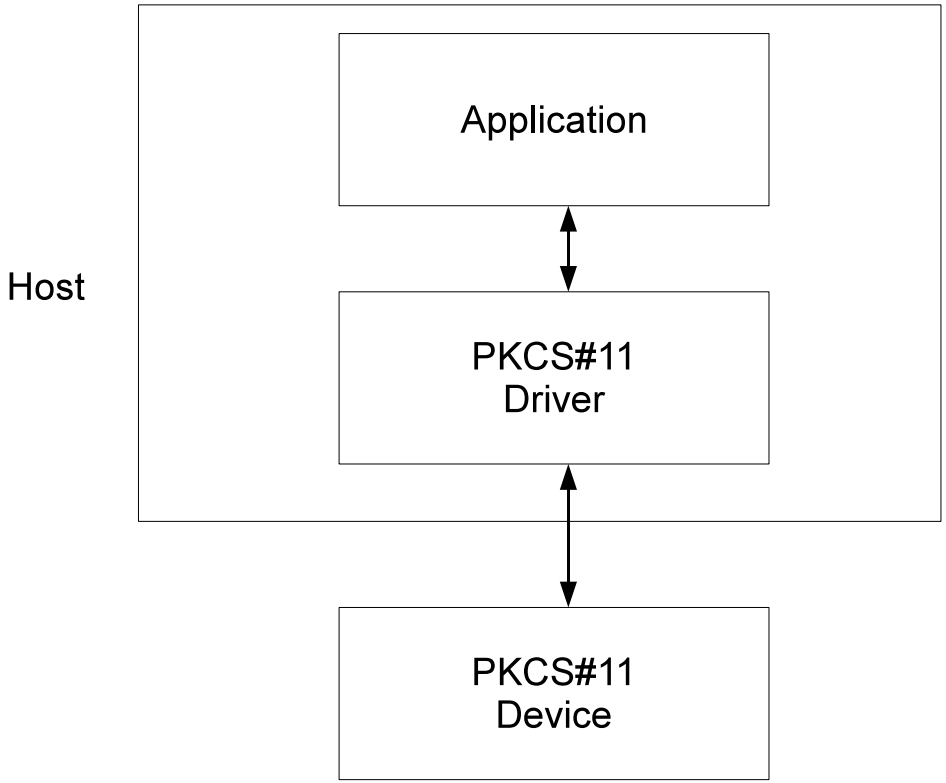
Cryptographic hardware devices such as smartcards, HSMs, authentication tokens etc. must offer an interface to application programs (API).

This API is security critical: no matter what sequence of commands are called, some security properties should hold.

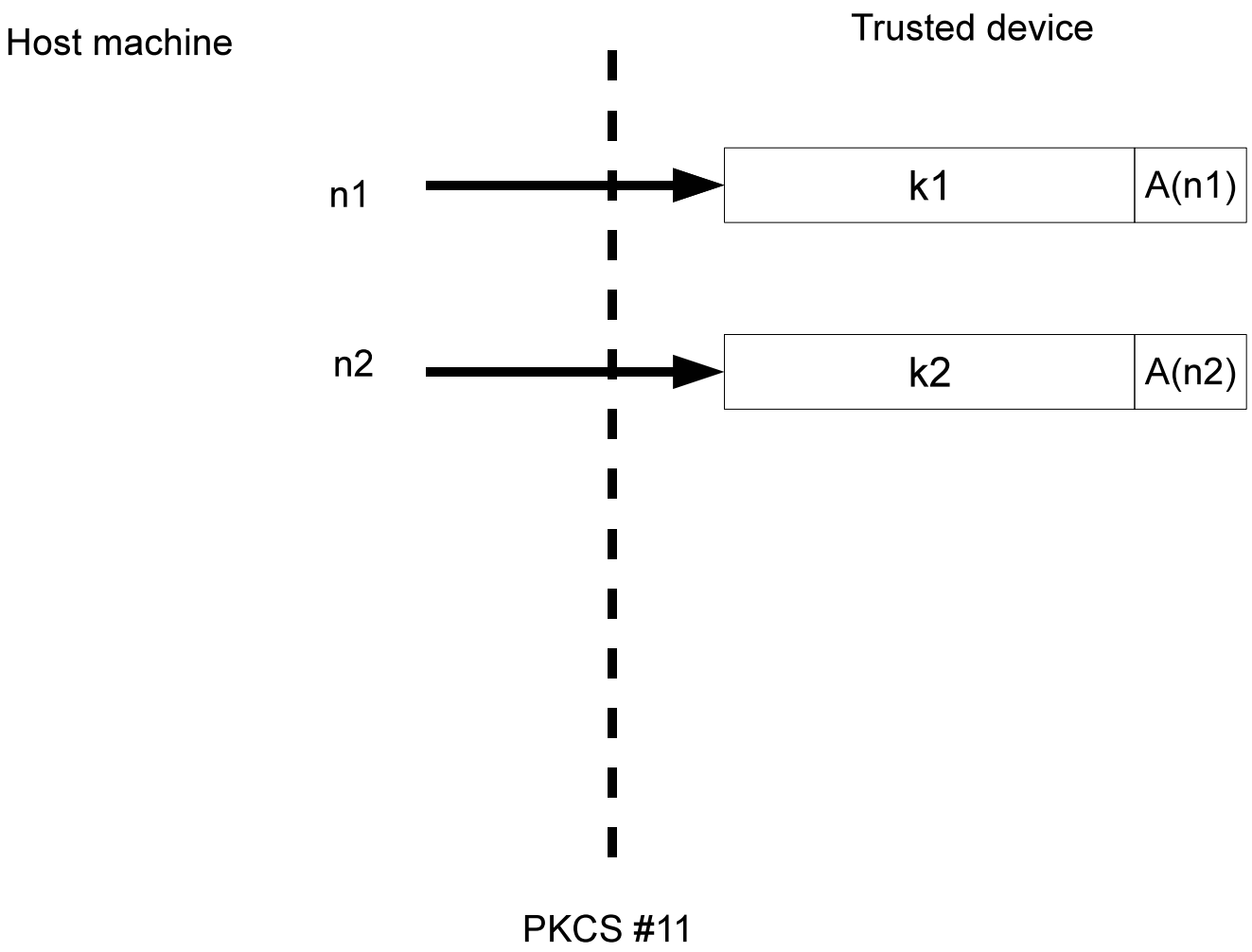
Designing such an interface is difficult: many vulnerabilities in deployed APIs have come to light.

For the last five years we have been researching the use of formal techniques to analyse such interfaces.

RSA PKCS#11



'Cryptoki' interface, v1.0 1995, v2.20 2004


Host machine
Trusted device
n1
k1
A(n1)
n2
k2
A(n2)
PKCS #11

Generating keys

A *key template* is a partial specification of key attributes
Used for creating, manipulating, and searching for objects

C_GenerateKey :

$$\mathcal{T} \rightarrow \mathsf{h}(\mathsf{n}, \mathsf{k}); \mathsf{T}$$

Setting Key Attributes

C_SetAttributeValue :

$$\mathcal{T}, \mathsf{h}(\mathsf{n}, \mathsf{k}) \quad \rightarrow \quad \mathsf{h}(\mathsf{n}, \mathsf{k}); \mathsf{T}$$

$\mathcal{T}$ can specify new values for any attributes, but may cause `CKR_TEMPLATE_INCONSISTENT`, `CKR_ATTRIBUTE_READ_ONLY`

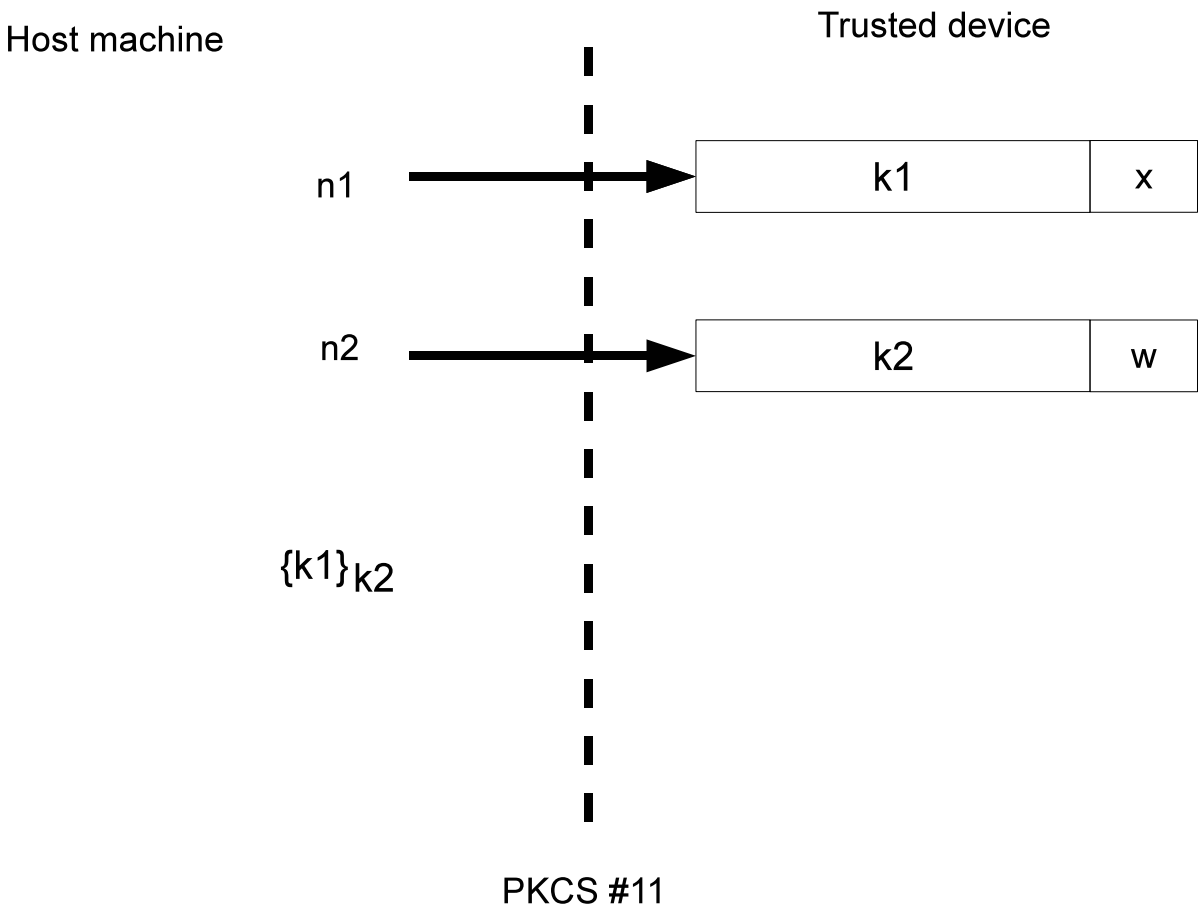
Wrap and Unwrap

Wrap :

$$h(x_1, y_1), h(x_2, y_2);\ \mathsf{wrap}(x_1), \mathsf{extract}(x_2) \rightarrow \{y_2\}_{y_1}$$

Unwrap :

$$h(x_2, y_2), \{y_1\}_{y_2}, \mathcal{T};\ \mathsf{unwrap}(x_2) \rightarrow h(n_1, y_1);\ \mathsf{extract}(n_1),\ \mathsf{T}$$


Host machine
Trusted device
n1
k1
x
n2
k2
w
{k1}k2
PKCS #11

Key Usage

Encrypt :

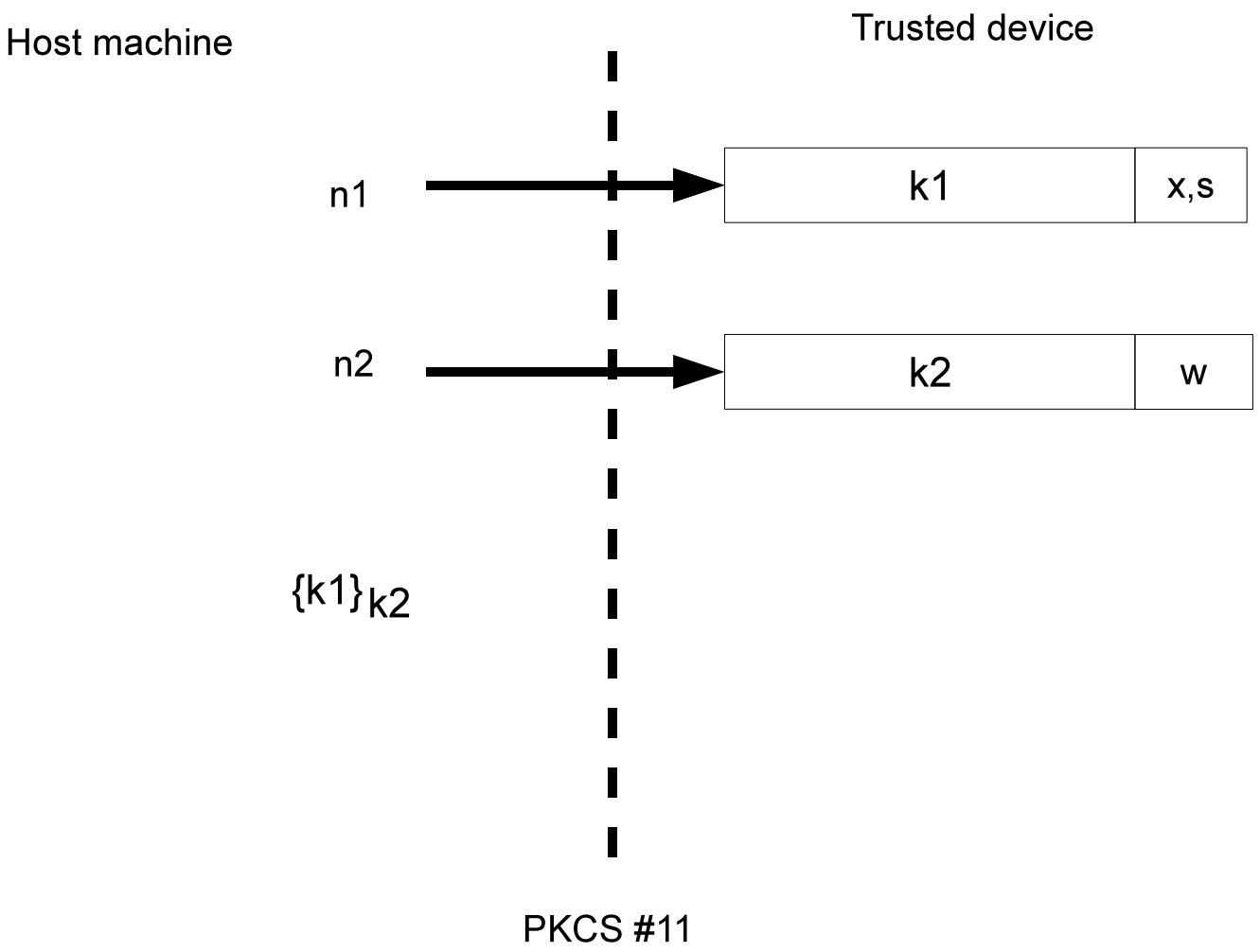
$$h(x_1, y_1), y_2;\ \text{encrypt}(x_1) \quad \rightarrow \quad \{y_2\}_{y_1}$$

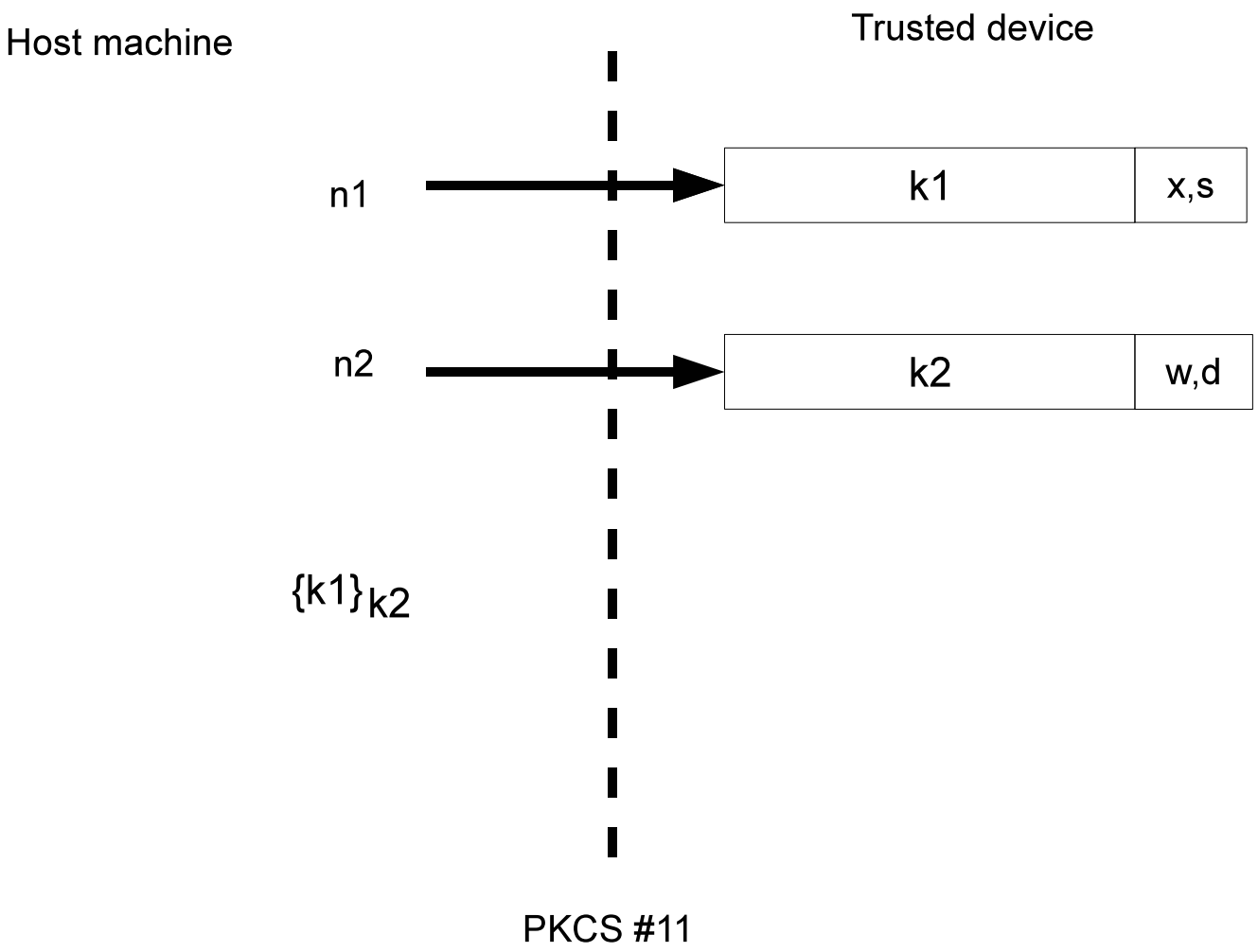
Decrypt :

$$h(x_1, y_1), \{y_2\}_{y_1};\ \text{decrypt}(x_1) \quad \rightarrow \quad y_2$$

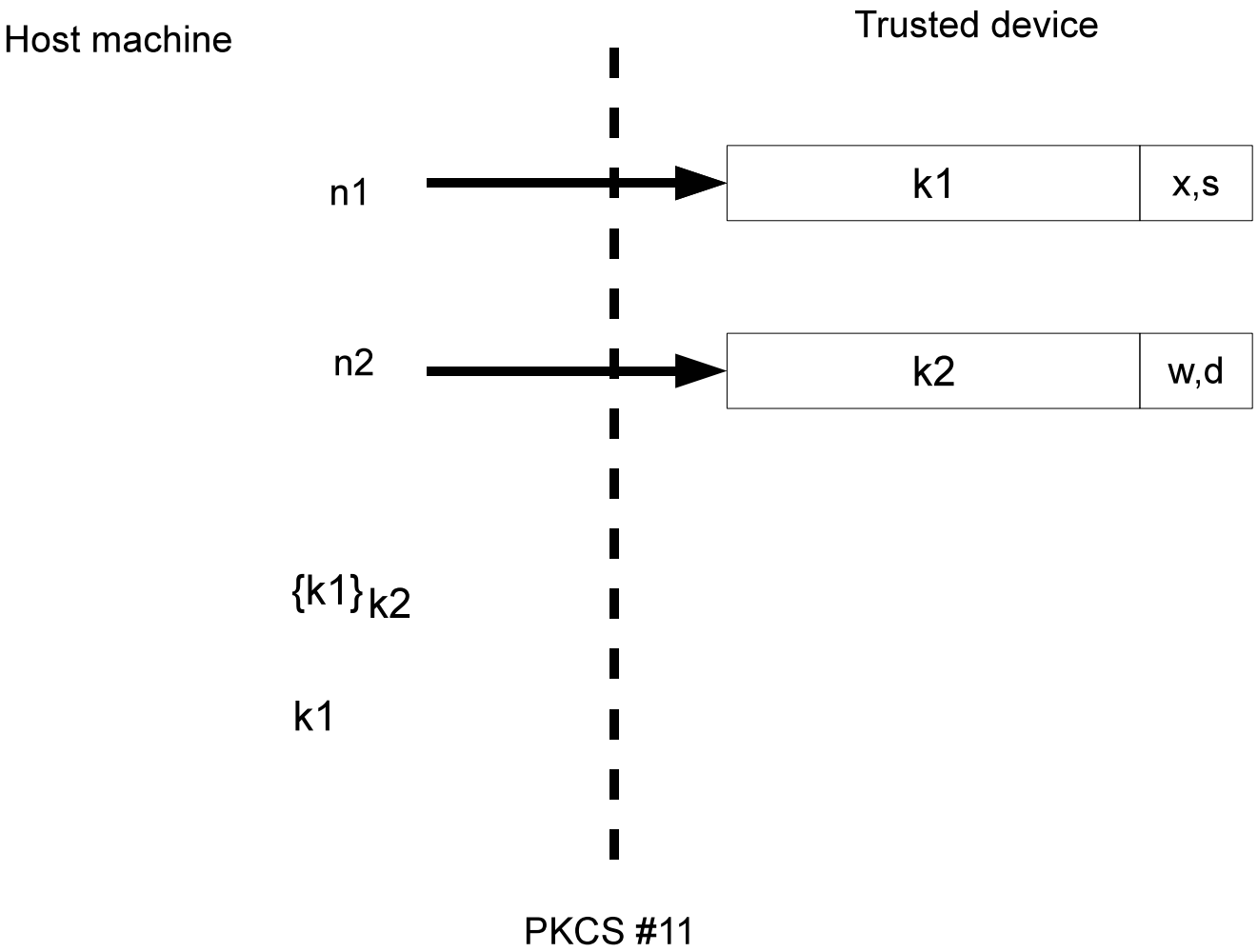
PKCS#11 Security

Section 7 of standard:
"1. Access to private objects on the token, and possibly to cryptographic functions and/or certificates on the token as well, requires a PIN.
2. Additional protection can be given to private keys and secret keys by marking them as "sensitive" or "unextractable". Sensitive keys cannot be revealed in plaintext off the token, and unextractable keys cannot be revealed off the token even when encrypted"
"Rogue applications and devices may also change the commands sent to the cryptographic device to obtain services other than what the application requested [but cannot] compromise keys marked "sensitive," since a key that is sensitive will always remain sensitive. Similarly, a key that is unextractable cannot be modified to be extractable."


Host machine
Trusted device
n1
k1
x,s
n2
k2
w
{k1}k2
PKCS #11


Host machine
Trusted device
n1
k1
x,s
n2
k2
w,d
{k1}k2
PKCS #11

Clulow, CHES 2003

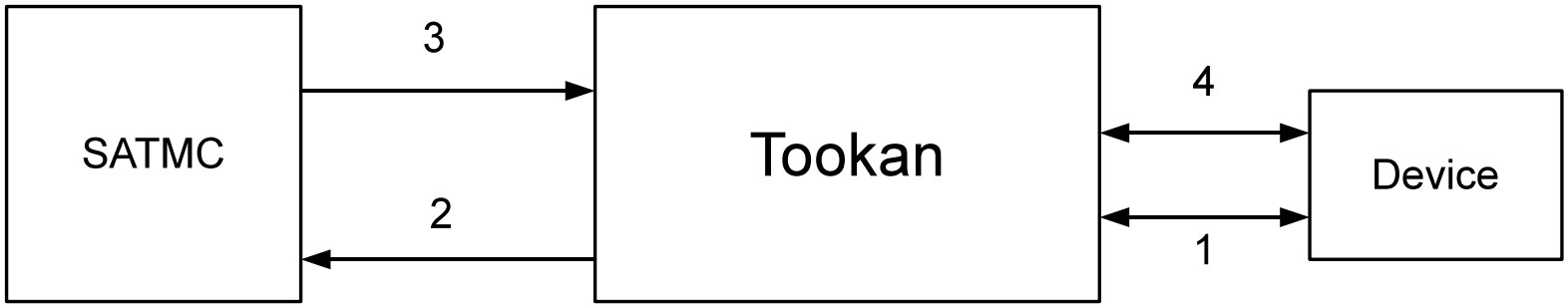


Prevent a key from doing decrypt and wrap..

Set_wrap:	$h(n_2, k_2)$	$\rightarrow$	$; wrap(n_2)$
Set_wrap:	$h(n_1, k_1)$	$\rightarrow$	$; wrap(n_1)$
Wrap:	$h(n_1, k_1), h(n_2, k_2)$	$\rightarrow$	$\{k_2\}_{k_1}$
Set_unwrap:	$h(n_1, k_1)$	$\rightarrow$	$; unwrap(n_1)$
Unwrap:	$h(n_1, k_1), \{k_2\}_{k_1}$	$\rightarrow$	$h(n_3, k_2)$
Wrap:	$h(n_2, k_2), h(n_1, k_1)$	$\rightarrow$	$\{k_1\}_{k_2}$
Set_decrypt:	$h(n_3, k_2)$	$\rightarrow$	$; decrypt(n_3)$
Decrypt:	$h(n_3, k_2), \{k_1\}_{k_2}$	$\rightarrow$	k_1

TOOKAN

'Tool for cryptoKi Analysis'



http://tookan.inria.gforge.fr/


TOOKAN

Brand	Device Model	Supported Functionality s	as	cobj	chan	w	ws	Attacks found wd	rs	ru	su	Tk
Aladdin	eToken PRO	✓	✓	✓	✓	✓	✓	✓				wd
Athena	ASEKey	✓	✓	✓								
Bull	Trustway RCI	✓	✓	✓	✓	✓	✓	✓				wd
Eutron	Crypto Id. ITSEC		✓	✓								
Feitian	StorePass2000	✓	✓	✓	✓	✓	✓	✓	✓	✓		rs
Feitian	ePass2000	✓	✓	✓	✓	✓	✓	✓	✓	✓		rs
Feitian	ePass3003Auto	✓	✓	✓	✓	✓	✓	✓	✓	✓		rs
Gemalto	SEG		✓		✓							
MXI	Stealth MXP Bio	✓	✓		✓							
RSA	SecurID 800	✓	✓	✓	✓				✓	✓	✓	rs
SafeNet	iKey 2032	✓	✓	✓		✓						
Sata	DKey	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	rs
ACS	ACOS5	✓	✓	✓	✓							
Athena	ASE Smartcard	✓	✓	✓								
Gemalto	Cyberflex V2	✓	✓	✓		✓	✓	✓				wd
Gemalto	SafeSite V1		✓		✓							
Gemalto	SafeSite V2	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	rs
Siemens	CardOS V4.3 B	✓	✓	✓		✓				✓		ru

Manufacturer Reaction

All were notified at least 5 months before publication.
We offered to publish responses on project website

RSA sent response, registered vulnerability with Mitre (CVE-2010-3321), issued security advisory 6 Oct 2010
Aladdin (now Safenet) and Gemalto sent a response for website

Minimal response from anyone else (e.g. requests to know who else is vulnerable)

Tookan now used by Boeing and a major UK-based bank.

Padding Oracles

A padding oracle returns true just when a ciphertext contains a correctly padded plaintext.

Padding oracle attacks send a number of chosen ciphertexts to the oracle to reveal the original plaintext.

Tookan detects these oracles using the `C_UnwrapKey` function - attack here reveals the imported key.

Asymmetric case (RSA PKCS#1.5) - make Bleichenbacher's 'Million message attack' in 15 000 messages (our paper at CRYPTO '12).

In the symmetric case (CBC-PKCS#5) attacks are already highly efficient.

Improvements to the Million Message Attack

Want to attack ciphertext c and discover $m = c^d \ mod \ n$

Choose integers s, send $c' = c \cdot s^e \ mod \ n$, to the padding oracle.

We showed that much can be learned about the plaintext by sending $c' = c \cdot u^e \cdot t^{-e}$

This allows us to search for s values much more efficiently: factor ten improvement in median over original algorithm

Ongoing Developments

- Executable attacks in C
- Man-in-the-middle analysis of PKCS#11 use
- Verification of fixes (PKCS#11 v2.2, 2.3, ACLs,..)
- Reverse-engineering at driver level (CCID, PKCS#15)
- Other APIs (Thales 8000, MSCAPI, Minidriver, Java APIs,..)

What Role for Formal Analysis of APIs?

Currently interfaces are not part of e.g. FIPS certification.
Many devices for which Tookan found vulnerabilities have CC certifications.
Formal tools like Tookan make analysis of interfaces practical, in particular because devices such as HSMs have rich configuration languages, not one static API.
Need to be able to state a policy, check the policy is what we want and check the device implements that policy.
Perhaps NIST Key Management Device standard will help?

`tookan.gforge.inria.fr`

`@TookanTool`