Analyse des API de sécurité

Graham Steel

Conférences de rentrée 2009

First Half

- What is a Security API
- Crypto 101
- Use of APIs in the cash machine network
- Attacks on the Visa security module
- Formal modelling


Infineon
TPM

Security APIs

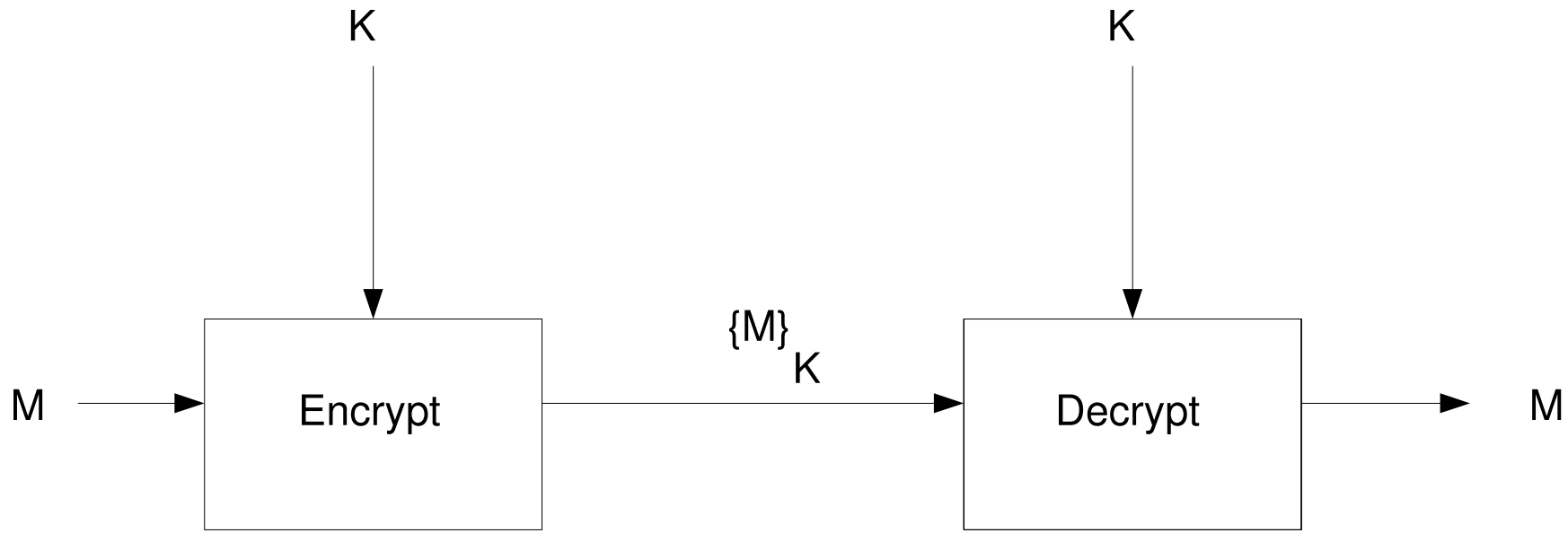
Host machine

Trusted device

Security API

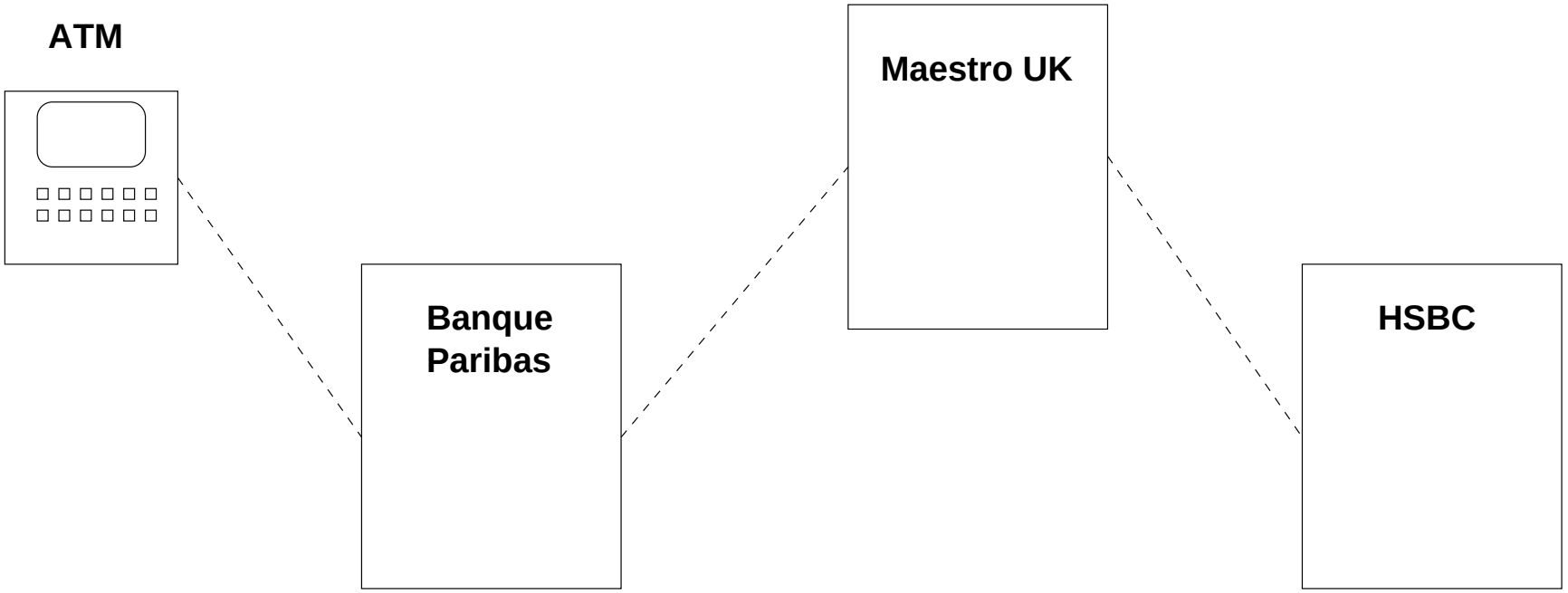
Crypto Basics

We consider only symmetric key crypto



Problem is now the security of key K

Cash Machine Network



HSMs

- Manufacturers include IBM, VISA, nCipher, Thales, Utimaco, HP
- Cost around $10 000

Master Key Scheme

Host machine

$\{ TMK1 \}_{KM}$

$\{ PDK1 \}_{KM}$

HSM

KM

A Word About Your PIN

IPIN derived by:

Write account number (PAN) as 0000AAAAAAAAAAAA

A Word About Your PIN

IPIN derived by:

Write account number (PAN) as 0000AAAAAAAAAAAA

3DES encrypt under a PDK (PIN Derivation Key)

A Word About Your PIN

IPIN derived by:

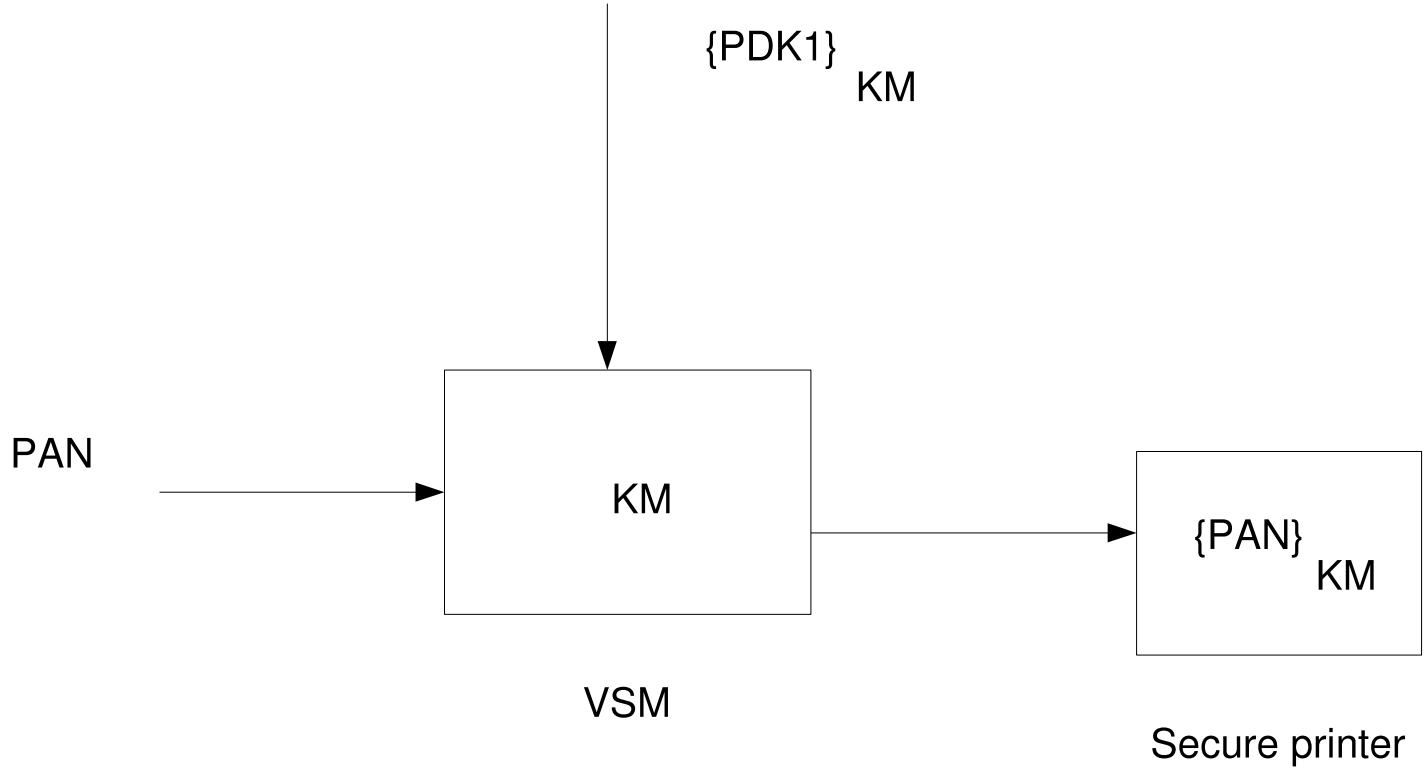
Write account number (PAN) as 0000AAAAAAAAAAAA

3DES encrypt under a PDK (PIN Derivation Key)

PIN = IPIN + Offset (modulo 10 each digit)

Offset NOT secure!

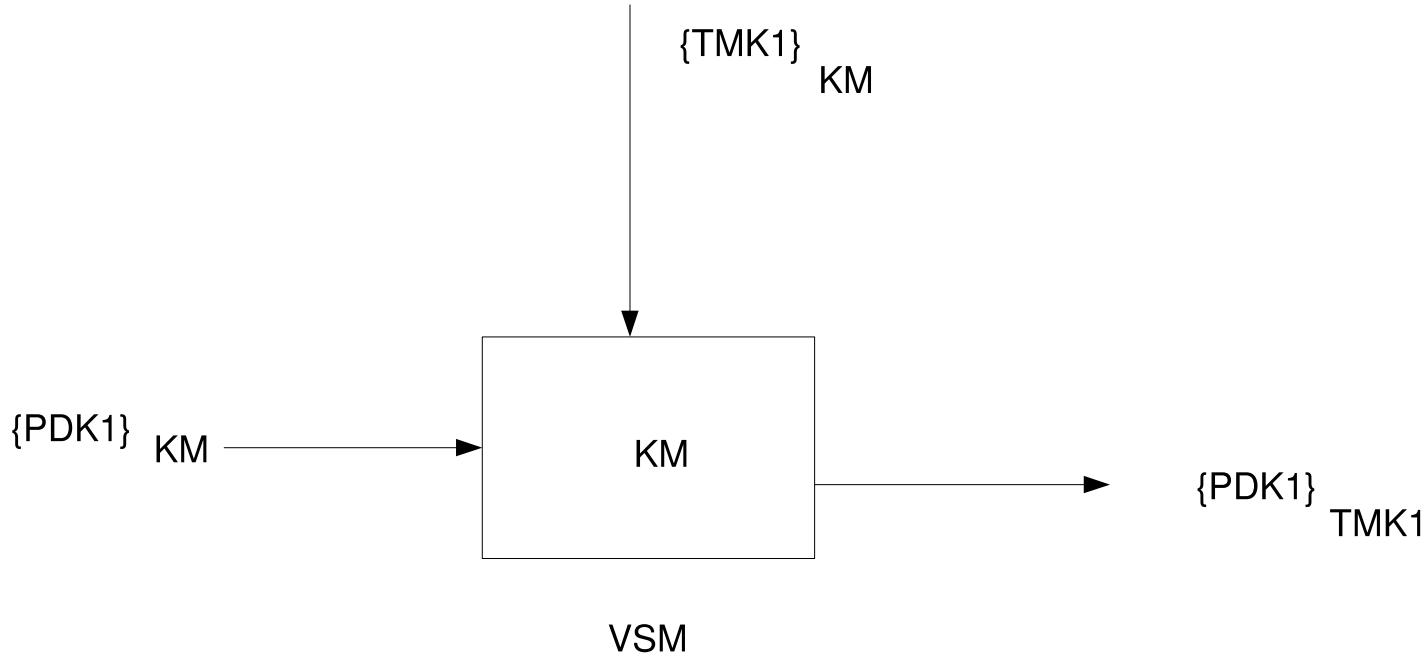
Example: Print Customer PIN



Host $\rightarrow$ HSM : PAN, $\{ \text{PDK1} \}_{\text{Km}}$

HSM $\rightarrow$ Printer : $\{ \text{PAN} \}_{\text{PDK1}}$

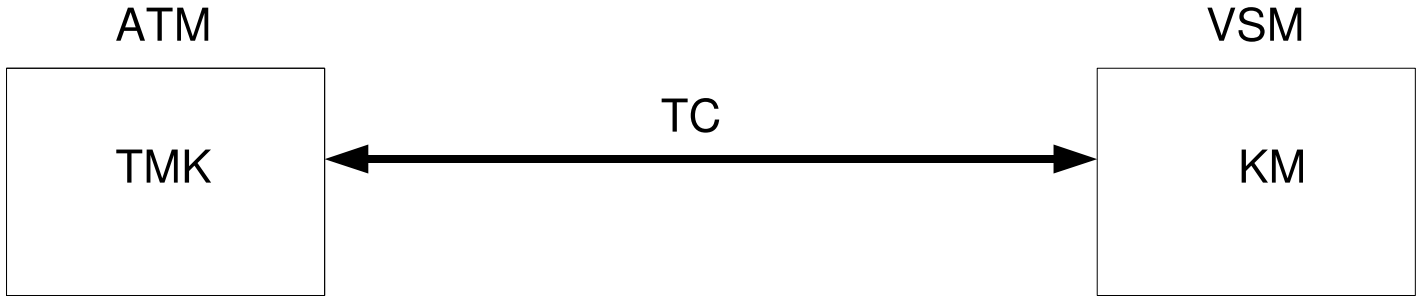
Example: Send PDK to Terminal



$$\text{Host} \rightarrow \text{HSM} : \{\text{PDK1}\}_{\text{Km}}, \{\text{TMK1}\}_{\text{Km}}$$

$$\text{HSM} \rightarrow \text{Host} : \{\text{PDK1}\}_{\text{TMK1}}$$

Terminal Comms Key



Managing Key Types

Host machine

VSM

$\{ TMK1 \}_{KM}$

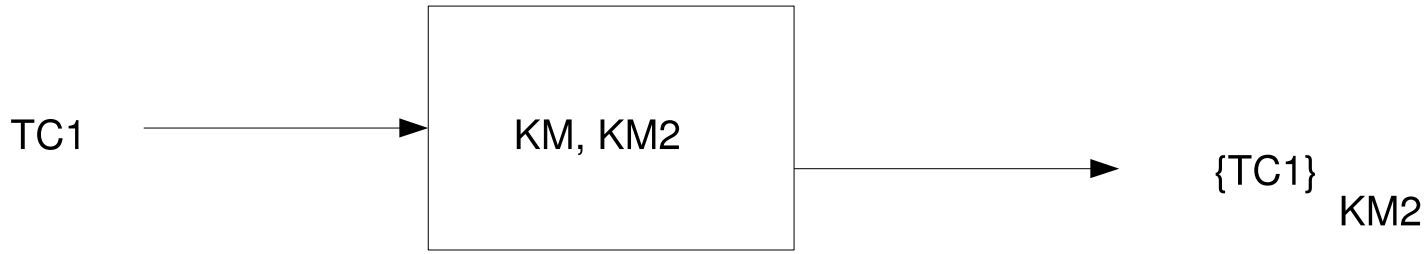
KM

$\{ PDK1 \}_{KM}$

$\{ TC1 \}_{KM2}$

KM2

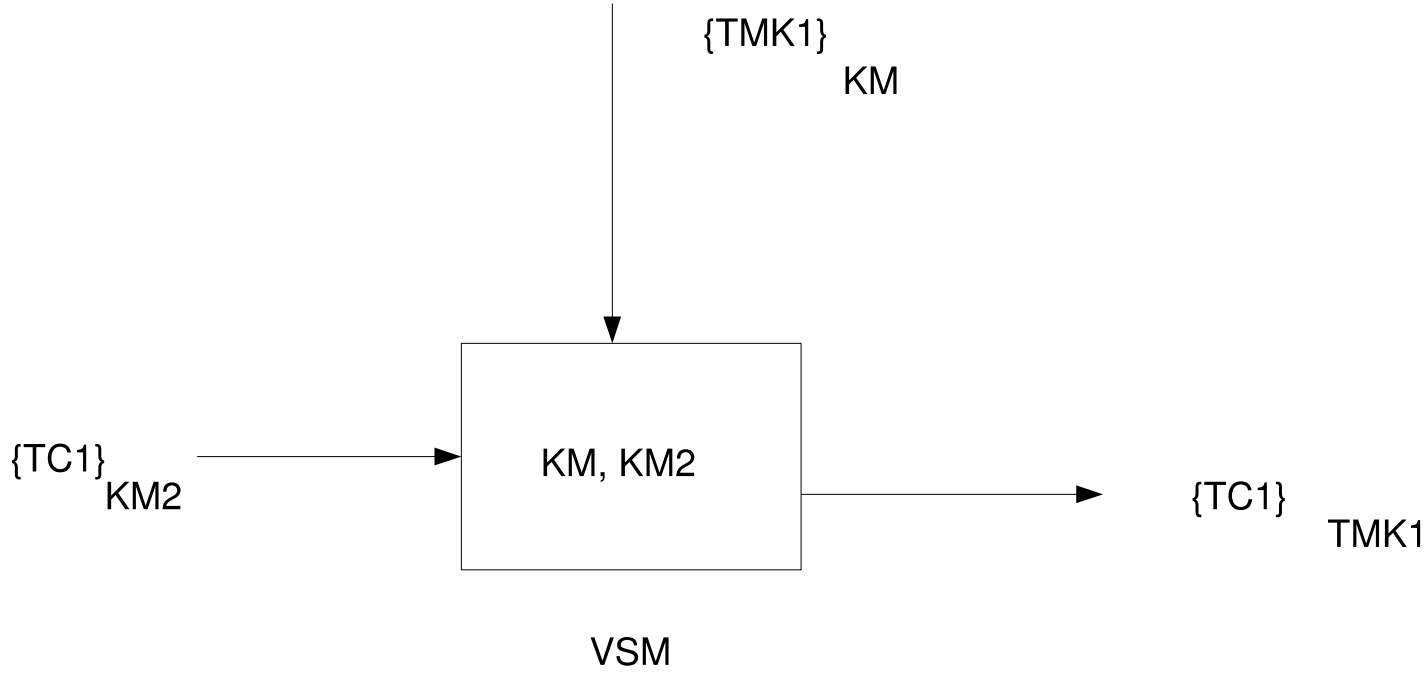
Example: Enter TC key



Host $\rightarrow$ HSM : TC

HSM $\rightarrow$ Host : $\{ \text{TC} \}_{\text{Km2}}$

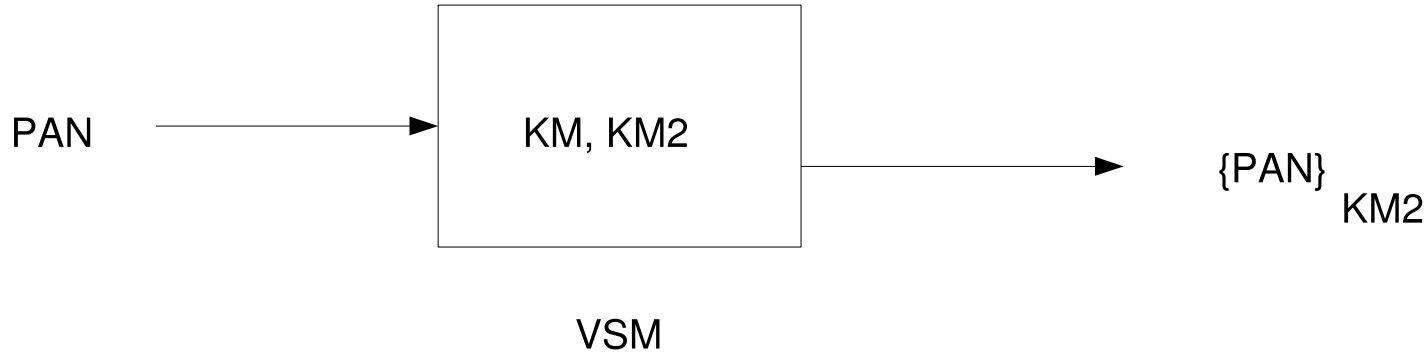
Example: Send TC to Terminal



$$\text{Host} \rightarrow \text{HSM} : \{ \text{TC} \}_{\text{Km2}}, \{ \text{TMK1} \}_{\text{Km}}$$

$$\text{HSM} \rightarrow \text{Host} : \{ \text{TC} \}_{\text{TMK1}}$$

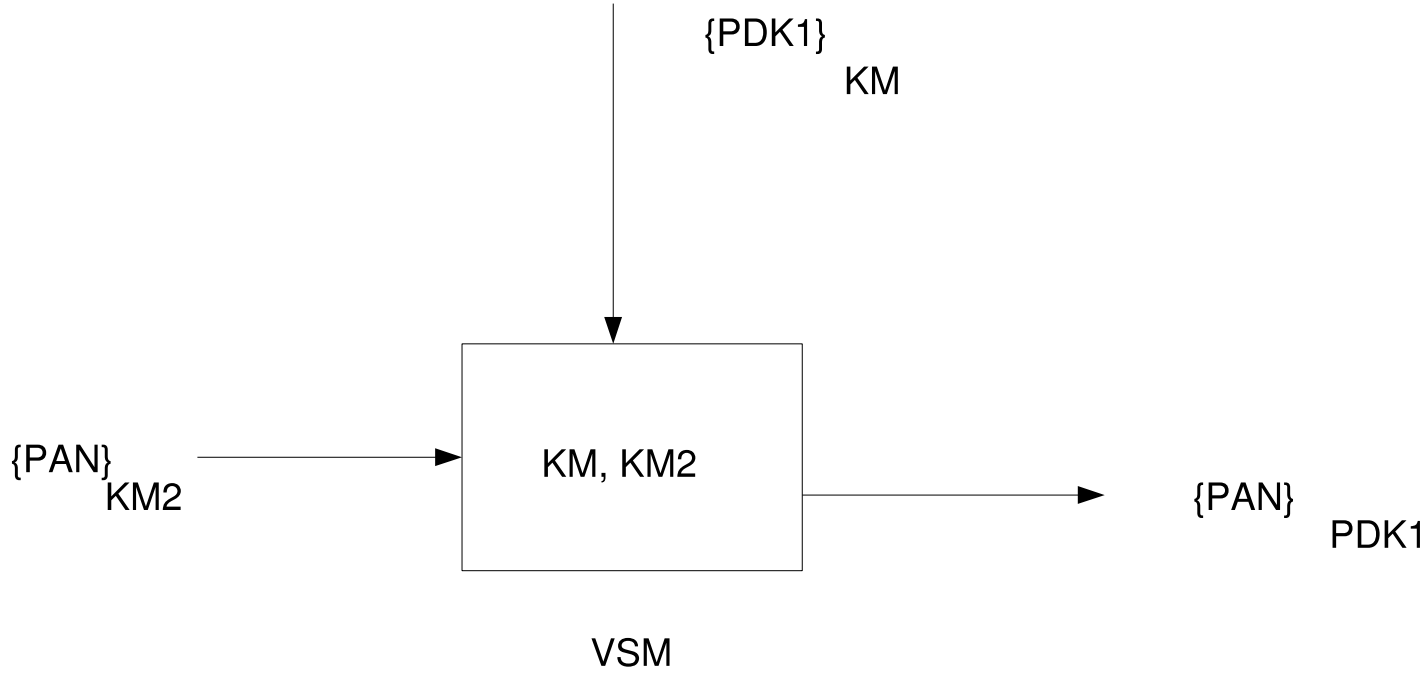
Attack - Step 1



$$\text{Spy} \rightarrow \text{HSM} : \text{PAN}$$

$$\text{HSM} \rightarrow \text{Spy} : \{ \text{PAN} \}_{Km2}$$

Attack - Step 2



$$\text{Spy} \rightarrow \text{HSM} : \{\text{PAN}\}_{Km2}, \{\text{PDK1}\}_{Km}$$

$$\text{HSM} \rightarrow \text{Host} : \{\text{PAN}\}_{PDK1}$$

Dolev-Yao Modelling

Bitstrings modelled as terms in an abstract algebra

Cryptographic algorithms modelled as function on terms

Attack modelled as derivation of secret term

Dolev-Yao Modelling 2

Atomic terms: $\mathsf{pdk1}, \mathsf{km}, \mathsf{km2}, \mathsf{pan}, \ldots$

Functions: $\{.\}.$

Intruder rules:

e.g. $\mathsf{x}, \mathsf{y} \rightarrow \{\mathsf{x}\}_{\mathsf{y}}$

API rules:

e.g. $\{\mathsf{x}\}_{\mathsf{km}}, \{\mathsf{y}\}_{\mathsf{km}} \rightarrow \{\mathsf{x}\}_{\mathsf{y}}$

Attack Search

Start with ‘initial knowledge’ of intruder

e.g. $\mathsf{pan}, \{\mathsf{pdk1}\}_{\mathsf{km}}, \{\mathsf{tmk1}\}_{\mathsf{km}}$

apply rules

Note: intruder knowledge increases monotonically, no need to backtrack in search

Goal is to derive term $\{\mathsf{pan}\}_{\mathsf{pdk1}}$

Can automate search with model checker, theorem prover, or custom tool

Example - using First Order Theorem Provers

API and intruder modelled in 13 FOL rules (Horn clauses), 12 terms in initial knowledge

Appears as problem SWV237 (www.tptp.org)

Attack found in $<$ 1 sec by several provers (Vampire, E,. . .)

VSM now has clear TC entry instruction disabled - problem SWV238

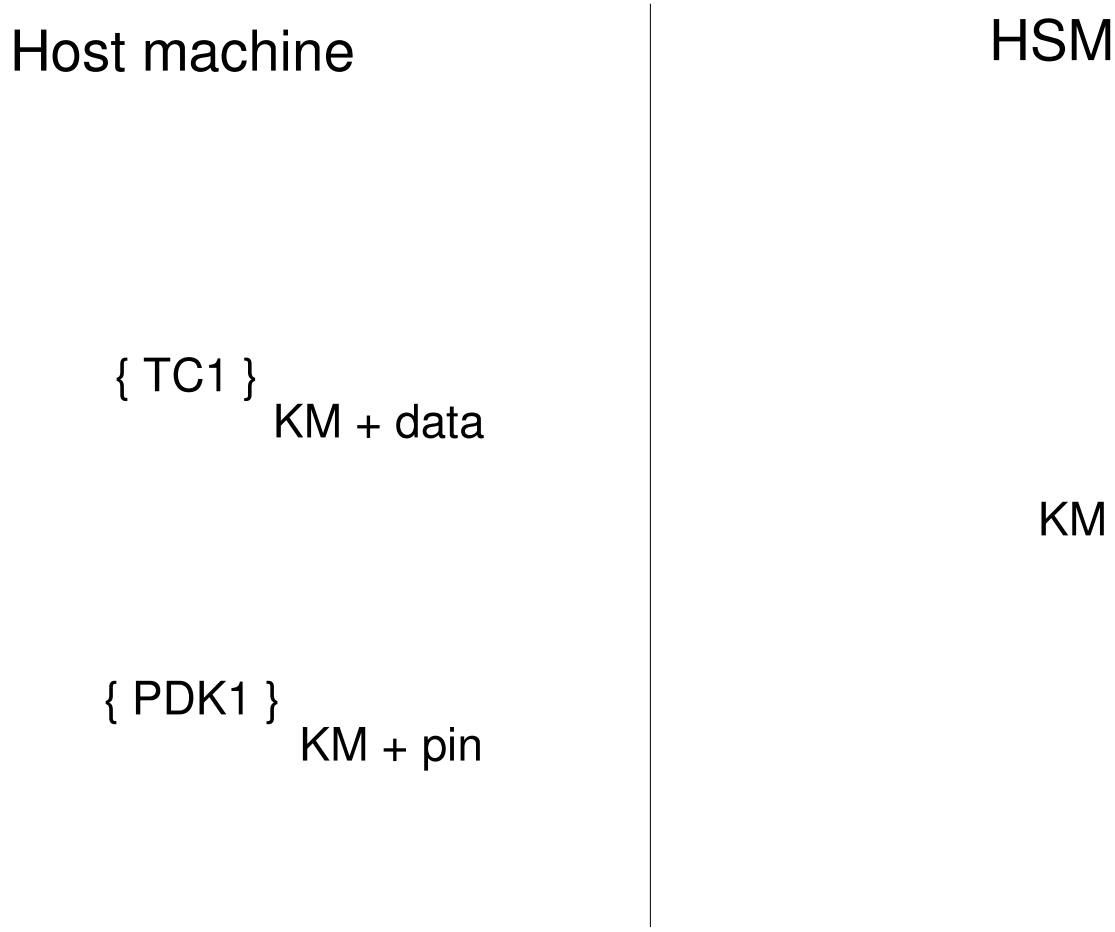
Only E can find a model (problem has no finite models)

Next half

- A much more complex API - IBM 4758 CCA
- Modelling XOR
- More attacks
- Proving decidability of security for certain APIs
- PKCS#11

IBM 4758 CCA API

CCA Types



CCA API - Examples

Encrypt Data:

$$\text{Host} \rightarrow \text{HSM} : \{ \text{d1} \}_{\text{km} \oplus \text{data}}, \text{message}$$

$$\text{HSM} \rightarrow \text{Host} : \{ \text{message} \}_{\text{d1}}$$

CCA API - Examples

Encrypt Data:

Host $\rightarrow$ HSM : $\{ \text{d1} \}_{\text{km} \oplus \text{data}}$, message

HSM $\rightarrow$ Host : $\{ \text{message} \}_{\text{d1}}$

Verify PIN:

Host $\rightarrow$ HSM : $\{ \text{PINBlock} \}_{\text{p1}}$, PAN, $\{ \text{pdk1} \}_{\text{km} \oplus \text{pin}}$,
OFFSET, $\{ \text{p1} \}_{\text{km} \oplus \text{ipinenc}}$

HSM $\rightarrow$ Host : yes/no

Importing Key Parts

‘Separation of duty’

Key k = k1 $\oplus$ k2

Host	$\rightarrow$	HSM	:	k1, TYPE
HSM	$\rightarrow$	Host	:	$\{ \text{k1} \}_{\text{km} \oplus \text{kp} \oplus \text{TYPE}}$

Importing Key Parts

‘Separation of duty’

Key k = k1 $\oplus$ k2

Host	$\rightarrow$	HSM	:	k1, TYPE
HSM	$\rightarrow$	Host	:	$\{ \text{k1} \}_{\text{km} \oplus \text{kp} \oplus \text{TYPE}}$

Host	$\rightarrow$	HSM	:	$\{ \text{k1} \}_{\text{km} \oplus \text{kp} \oplus \text{TYPE}}$, k2, TYPE
HSM	$\rightarrow$	Host	:	$\{ \text{k1} \oplus \text{k2} \}_{\text{km} \oplus \text{TYPE}}$

Usually used to import a ‘key encrypting key’ ($\{ \text{KEK} \}_{\text{km} \oplus \text{imp}}$)

Importing Encrypted Keys

Exported from another 4758 encrypted under $\mathsf{KEK} \oplus \mathsf{TYPE}$

Key Import:

$$\mathsf{Host} \rightarrow \mathsf{HSM} : \{\, \mathsf{KEY1} \,\}_{\mathsf{KEK} \oplus \mathsf{TYPE}}, \mathsf{TYPE}, \{\, \mathsf{KEK} \,\}_{\mathsf{km} \oplus \mathsf{imp}}$$

$$\mathsf{HSM} \rightarrow \mathsf{Host} : \{\, \mathsf{KEY1} \,\}_{\mathsf{km} \oplus \mathsf{TYPE}}$$

Attack (Bond, 2001) (part 1)

PIN derivation key: $\{ \text{pdk} \}_{\text{kek} \oplus \text{pin}}$

Have key part $\{ \text{kek} \oplus \text{k2} \}_{\text{km} \oplus \text{imp} \oplus \text{kp}}$ for known k2

Attack (Bond, 2001) (part 1)

PIN derivation key: $\{ \mathsf{pdk} \}_{\mathsf{kek} \oplus \mathsf{pin}}$

Have key part $\{ \mathsf{kek} \oplus \mathsf{k2} \}_{\mathsf{km} \oplus \mathsf{imp} \oplus \mathsf{kp}}$ for known k2

$$\mathsf{Host} \rightarrow \mathsf{HSM} : \{ \mathsf{kek} \oplus \mathsf{k2} \}_{\mathsf{km} \oplus \mathsf{kp} \oplus \mathsf{imp}}, \mathsf{k2} \oplus \mathsf{pin} \oplus \mathsf{data}, \mathsf{imp}$$

$$\mathsf{HSM} \rightarrow \mathsf{Host} : \{\mathsf{kek} \oplus \mathsf{pin} \oplus \mathsf{data}\}_{\mathsf{km} \oplus \mathsf{imp}}$$

Attack (Bond, 2001) (part 2)

Key Import

$$\text{Host} \rightarrow \text{HSM} : \{ \text{pdk} \}_{\text{kek} \oplus \text{pin}}, \text{data}, \{ \text{kek} \oplus \text{pin} \oplus \text{data} \}_{\text{km} \oplus \text{imp}}$$

$$\text{HSM} \rightarrow \text{Host} : \{ \text{pdk} \}_{\text{km} \oplus \text{data}}$$

Attack (Bond, 2001) (part 2)

Key Import

$$\text{Host} \rightarrow \text{HSM} : \{ \text{pdk} \}_{\text{kek} \oplus \text{pin}}, \text{data}, \{ \text{kek} \oplus \text{pin} \oplus \text{data} \}_{\text{km} \oplus \text{imp}}$$

$$\text{HSM} \rightarrow \text{Host} : \{ \text{pdk} \}_{\text{km} \oplus \text{data}}$$

Encrypt data

$$\text{Host} \rightarrow \text{HSM} : \{ \text{pdk} \}_{\text{km} \oplus \text{data}}, \text{pan}$$

$$\text{HSM} \rightarrow \text{Host} : \{ \text{pan} \}_{\text{pdk}} \text{ (= PIN!)}$$

Formal Modelling

Encrypt data:

$$x, \{d1\}_{km \oplus data} \quad \rightarrow \quad \{x\}_{d1}$$

Formal Modelling

Encrypt data:

$$x, \{d1\}_{km \oplus data} \rightarrow \{x\}_{d1}$$

Intruder rules:

$$\begin{aligned} x, y &\rightarrow \{x\}_y \\ \{x\}_y, y &\rightarrow x \\ x, y &\rightarrow x \oplus y \end{aligned}$$

Formal Modelling

Encrypt data:

$$x, \{d1\}_{km \oplus data} \quad \rightarrow \quad \{x\}_{d1}$$

Intruder rules:

$$\begin{aligned} x, y &\rightarrow \{x\}_y \\ \{x\}_y, y &\rightarrow x \\ x, y &\rightarrow x \oplus y \end{aligned}$$

Also need

$$\begin{aligned} x \oplus (y \oplus z) &= (x \oplus y) \oplus z \\ x \oplus y &= y \oplus x \\ x \oplus x &= 0 \end{aligned}$$

Using TPTP

Problem SWV234 in TPTP

Only 11 rules, 15 items in knowledge set

However, only most recent version of Vampire can find the attack

‘Combinatorial explosion’ caused by XOR is the problem

Cannot verify fixes using TPTP provers (yet..)

Characterisation of Class

Finite set of atoms (km, imp, data, pin, . . .)

XOR term ::= atom

atom $\oplus$ XOR term

Encryption term ::= $\{$ XOR term $\}_{\text{XOR term}}$

Well Formed Term ::= Encryption term

XOR term

Well Formed Rule ::= WFT, . . ., WFT $\rightarrow$ WFT

Theorem

If:

R finite set of well-formed rules

S finite set of well-formed ground terms

u some ground well-formed term

Then:

$S \vdash_R u \iff S \vdash_R u$ using only well-formed terms.

Theorem

If:

R finite set of well-formed rules

S finite set of well-formed ground terms

u some ground well-formed term

Then:

$S \vdash_R u \iff S \vdash_R u$ using only well-formed terms.

Corollary:

The question of whether $S \vdash_R u$ is decidable

Decision Procedure

2^{12} possible unencrypted terms

2^{24} possible encrypted terms ($\{\ .\ \}_.$)

Decision Procedure

2^{12} possible unencrypted terms

2^{24} possible encrypted terms ($\{\ .\ \}_.$)

Encode terms as integers

$\mathsf{kek} \oplus \mathsf{pin} \oplus \mathtt{data}$	$\rightarrow$	km	kp	kek	imp	exp	data	pin
19	$\leftarrow$	0	0	1	0	0	1	1

Decision Procedure

2^{12} possible unencrypted terms

2^{24} possible encrypted terms ($\{\ .\ \}_.$)

Encode terms as integers

$\mathsf{kek} \oplus \mathsf{pin} \oplus \mathsf{data}$	$\rightarrow$	km	kp	kek	imp	exp	data	pin
19	$\leftarrow$	0	0	1	0	0	1	1

Each rule is a partial function $f : \mathbb{N}^k \rightarrow \mathbb{N}$ for k inputs

e.g. $f_1 : x1, x2 \rightarrow x1 \oplus x2 \quad \equiv \quad x1, x2 \rightarrow x1 \oplus x2$

Decision Procedure

2^{12} possible unencrypted terms

2^{24} possible encrypted terms ($\{\ .\ \}_.$)

Encode terms as integers

kek $\oplus$ pin $\oplus$ data	$\rightarrow$	km	kp	kek	imp	exp	data	pin
19	$\leftarrow$	0	0	1	0	0	1	1

Each rule is a partial function $f : \mathbb{N}^k \rightarrow \mathbb{N}$ for k inputs

e.g. $f_1 : x1, x2 \rightarrow x1 \oplus x2 \quad \equiv \quad x1, x2 \rightarrow x1 \oplus x2$

$$f_2 : [xkey|x], xtype, [xkek|q] \rightarrow [xkey|q \oplus xtype] \quad \text{IF} \quad x = xkek \oplus xtype$$

Decision Procedure

1. Allocate sufficient memory for all possible terms
2. Set to 1 locations corresponding to initial knowledge, rest to 0
3. Exhaustively apply each rule, setting newly discovered terms to 1
4. Repeat 3 until no new terms are discovered

Decision Procedure

1. Allocate sufficient memory for all possible terms
2. Set to 1 locations corresponding to initial knowledge, rest to 0
3. Exhaustively apply each rule, setting newly discovered terms to 1
4. Repeat 3 until no new terms are discovered

Some optimisations used

Details in [Cortier, Keighren, Steel, TACAS’07]

Decision Procedure

1. Allocate sufficient memory for all possible terms
2. Set to 1 locations corresponding to initial knowledge, rest to 0
3. Exhaustively apply each rule, setting newly discovered terms to 1
4. Repeat 3 until no new terms are discovered

Some optimisations used

Details in [Cortier, Keighren, Steel, TACAS’07]

Now we can verify fixes..

IBM Recommendations

Published in response to attacks

IBM Recommendations

Published in response to attacks

1. Use asymmetric key crypto for key import
   – 2 officer protocol to generate key pair at destination, transfer public key to source
   – PKA_SYMMETRIC_KEY_IMPORT command

IBM Recommendations

Published in response to attacks

1. Use asymmetric key crypto for key import

   – 2 officer protocol to generate key pair at destination, transfer public key to source

   – PKA_SYMMETRIC_KEY_IMPORT command

2. More access control

   – security officers access fewer commands

IBM Recommendations

Published in response to attacks

1. Use asymmetric key crypto for key import
   – 2 officer protocol to generate key pair at destination, transfer public key to source
   – PKA_SYMMETRIC_KEY_IMPORT command
2. More access control
   – security officers access fewer commands
3. Procedural controls to check entered key parts

IBM Recommendations

Published in response to attacks

1. Use asymmetric key crypto for key import

   – 2 officer protocol to generate key pair at destination, transfer public key to source

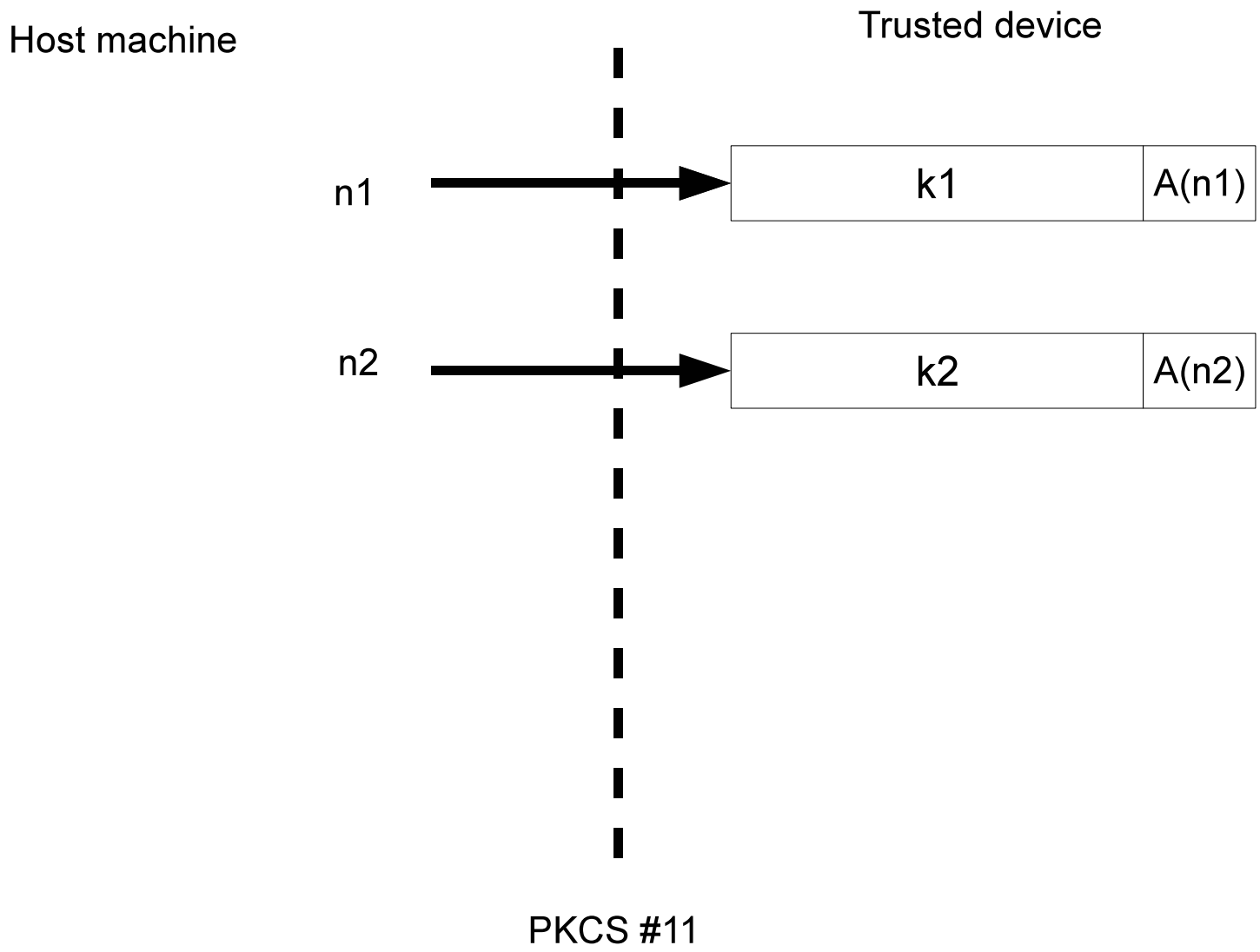
   – PKA_SYMMETRIC_KEY_IMPORT command

2. More access control

   – security officers access fewer commands

3. Procedural controls to check entered key parts

2 and 3 verified in a few seconds, but 1 has a simple attack (see [CKS 07])


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

Key Management - 1

$$\text{KeyGenerate} : \xrightarrow{\text{new n,k}} \quad h(n,k); L$$

Where $L = \neg\text{extractable}(n), \neg\text{wrap}(n), \neg\text{unwrap}(n),$
$\neg\text{encrypt}(n), \neg\text{decrypt}(n), \neg\text{sensitive}(n)$

Key Management - 2

Wrap :

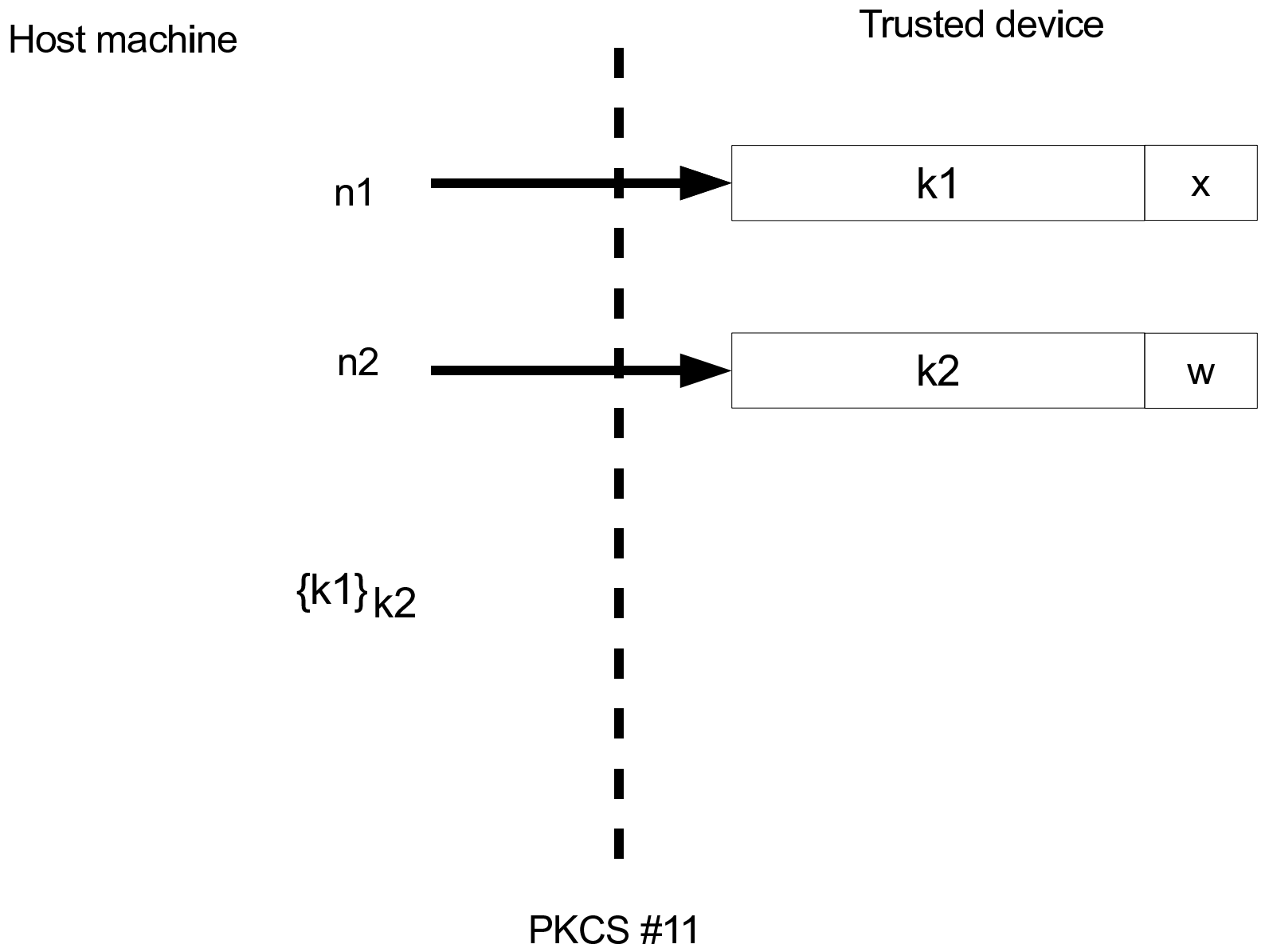
$$h(x_1, y_1), h(x_2, y_2);\ \mathsf{wrap}(x_1), \mathsf{extract}(x_2) \quad \rightarrow \quad \{y_2\}_{y_1}$$

Unwrap :

$$h(x_2, y_2), \{y_1\}_{y_2};\ \mathsf{unwrap}(x_2) \quad \xrightarrow{\text{new } n_1} \quad h(n_1, y_1);\ \mathsf{extract}(n_1),\ L$$

where $L =$

$\neg\mathsf{wrap}(n_1), \neg\mathsf{unwrap}(n_1), \neg\mathsf{encrypt}(n_1), \neg\mathsf{decrypt}(n_1), \neg\mathsf{sensitive}(n_1).$


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

Key Management - 3

$$
\begin{array}{lrcl}
\mathsf{Set\_Wrap}: & \mathsf{h}(x_1, y_1);\ \neg\mathsf{wrap}(x_1) & \rightarrow & ;\mathsf{wrap}(x_1) \\
\mathsf{Set\_Encrypt}: & \mathsf{h}(x_1, y_1);\ \neg\mathsf{encrypt}(x_1) & \rightarrow & ;\mathsf{encrypt}(x_1) \\
\vdots & & \vdots & \\
\mathsf{UnSet\_Wrap}: & \mathsf{h}(x_1, y_1);\ \mathsf{wrap}(x_1) & \rightarrow & ;\neg\mathsf{wrap}(x_1) \\
\mathsf{UnSet\_Encrypt}: & \mathsf{h}(x_1, y_1);\ \mathsf{encrypt}(x_1) & \rightarrow & ;\neg\mathsf{encrypt}(x_1) \\
\vdots & & \vdots &
\end{array}
$$

Some restrictions, e.g. can't unset sensitive

Key Usage

Encrypt :

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

Decrypt :

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

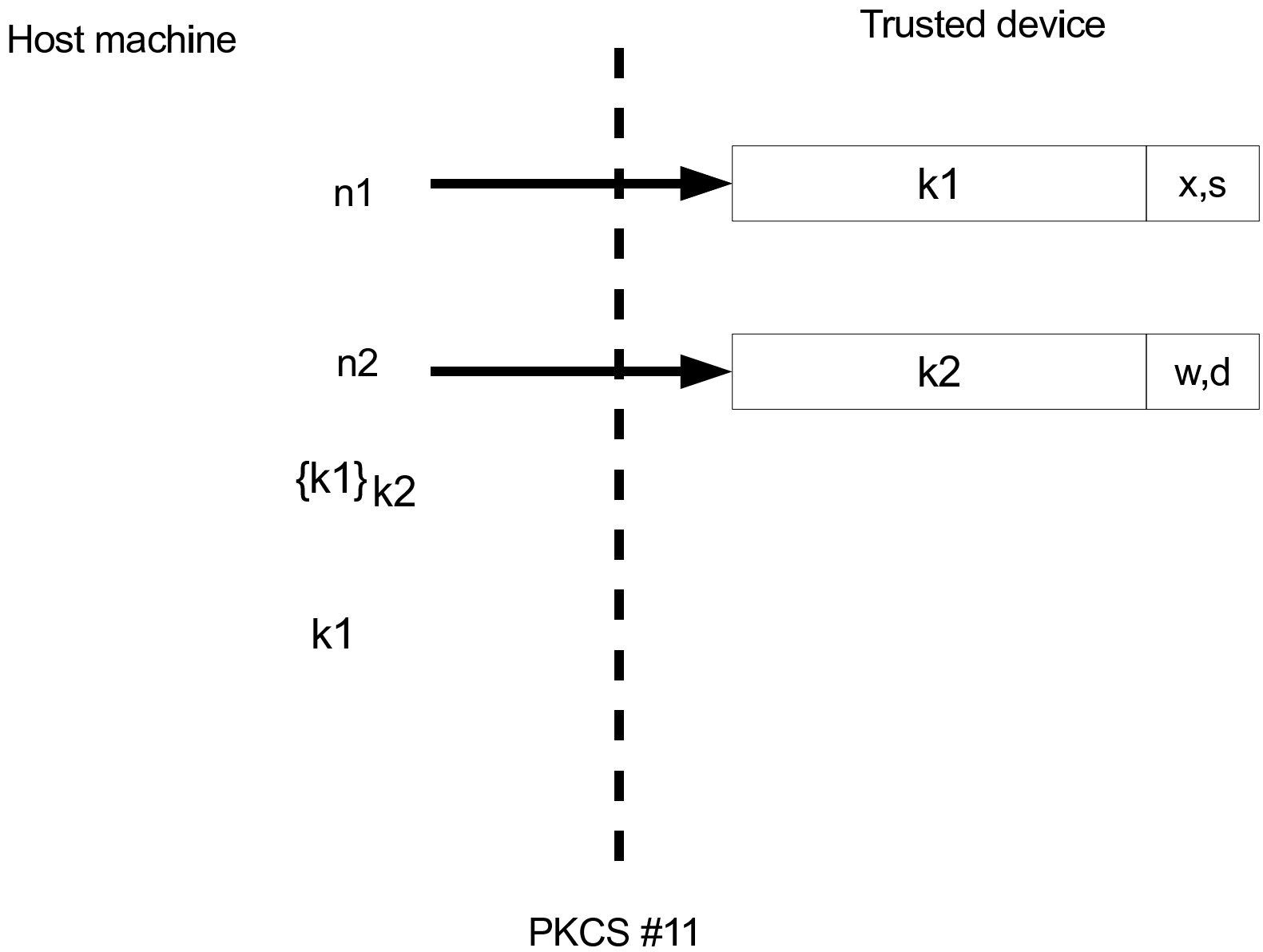
Key Separation Attack (Clulow, 2003)

Intruder knows: $h(n_1, k_1)$, $h(n_2, k_2)$.

State: $\mathsf{wrap}(n_2)$, $\mathsf{decrypt}(n_2)$, $\mathsf{sensitive}(n_1)$, $\mathsf{extract}(n_1)$

Wrap: $h(n_2, k_2), h(n_1, k_1) \rightarrow \{k_1\}_{k_2}$

Decrypt: $h(n_2, k_2), \{k_1\}_{k_2} \rightarrow k_1$


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

Formal Model

Rules:

$$T; L \xrightarrow{\mathsf{new}\,\tilde{n}} T'; L'$$

Formal Model

Rules:

$$T; L \xrightarrow{\mathsf{new}\,\tilde{n}} T'; L'$$

Important difference between this and previous and APIs:

State $L, L', L'', \ldots$ not monotonic, and may contain loops

Modes

$$
\begin{aligned}
\mathsf{h} &: \mathsf{Nonce} \times \mathsf{Key} \to \mathsf{Handle} \\
\mathsf{senc} &: \mathsf{Key} \times \mathsf{Key} \to \mathsf{Cipher} \\
\mathsf{aenc} &: \mathsf{Key} \times \mathsf{Key} \to \mathsf{Cipher} \\
\mathsf{pub} &: \mathsf{Seed} \to \mathsf{Key} \\
\mathsf{priv} &: \mathsf{Seed} \to \mathsf{Key} \\
\mathsf{a} &: \mathsf{Nonce} \to \mathsf{Attribute} \quad \text{for all } \mathsf{a} \in \mathcal{A} \\
\mathsf{x}_1, \mathsf{x}_2, \mathsf{n}_1, \mathsf{n}_2 &: \mathsf{Nonce} \\
\mathsf{y}_1, \mathsf{y}_2, \mathsf{k}_1, \mathsf{k}_2 &: \mathsf{Key} \\
\mathsf{z}, \mathsf{s} &: \mathsf{Seed}
\end{aligned}
$$

Implementation

- Bound fresh material

  finite vocabulary of terms for acyclic modes

- Propositional encoding

  one prop each $\mathsf{I}(\mathsf{term})$, $\mathsf{att}(\mathsf{nonce})$

- NuSMV

  search for $\mathsf{I}(\mathsf{k})$ for sensitive k

Fix decrypt/wrap attack..

Fix decrypt/wrap attack..

$$\text{Set\_Wrap}: \quad h(x_1, y_1);\ \neg\text{wrap}(x_1), \neg\text{decrypt}(x_1) \rightarrow \text{wrap}(x_1)$$

$$\text{Set\_Decrypt}: \quad h(x_1, y_1);\ \neg\text{wrap}(x_1), \neg\text{decrypt}(x_1) \rightarrow \text{decrypt}(x_1)$$

Fix decrypt/wrap attack..

$$\text{Set\_Wrap}: \quad h(x_1, y_1);\ \neg wrap(x_1), \neg decrypt(x_1) \rightarrow wrap(x_1)$$

$$\text{Set\_Decrypt}: \quad h(x_1, y_1);\ \neg wrap(x_1), \neg decrypt(x_1) \rightarrow decrypt(x_1)$$

Unset_Wrap

Unset_Decrypt

Another Attack

Intruder knows: $h(n_1, k_1)$, $h(n_2, k_2)$, k_3

State: $\mathsf{sensitive}(n_1), \mathsf{extract}(n_1)$, $\mathsf{unwrap}(n_2)$, $\mathsf{encrypt}(n_2)$

SEncrypt:	$h(n_2, k_2), k_3$	$\rightarrow$	$\mathsf{senc}(k_3, k_2)$
Unwrap:	$h(n_2, k_2), \mathsf{senc}(k_3, k_2)$	$\xrightarrow{\mathsf{new}\, n_3}$	$h(n_3, k_3)$
Set_wrap:	$h(n_3, k_3)$	$\rightarrow$	$\mathsf{wrap}(n_3)$
Wrap:	$h(n_3, k_3), h(n_1, k_1)$	$\rightarrow$	$\mathsf{senc}(k_1, k_3)$
Intruder:	$\mathsf{senc}(k_1, k_3), k_3$	$\rightarrow$	k_1

Fix decrypt/wrap, encrypt/unwrap..

Fix decrypt/wrap, encrypt/unwrap..

Intruder knows: $h(n_1, k_1)$, $h(n_2, k_2)$, k_3

State: $\mathsf{sensitive}(n_1), \mathsf{extract}(n_1)$, $\mathsf{extract}(n_2)$

Set_wrap:	$h(n_2, k_2)$	$\rightarrow$	$\mathsf{wrap}(n_2)$
Wrap:	$h(n_2, k_2), h(n_2, k_2)$	$\rightarrow$	$\mathsf{senc}(k_2, k_2)$
Set_unwrap:	$h(n_2, k_2)$	$\rightarrow$	$\mathsf{unwrap}(n_2)$
Unwrap:	$h(n_2, k_2), \mathsf{senc}(k_2, k_2)$	$\xrightarrow{\mathsf{new}\, n_4}$	$h(n_4, k_2)$
Wrap:	$h(n_2, k_2), h(n_1, k_1)$	$\rightarrow$	$\mathsf{senc}(k_1, k_2)$
Set_decrypt:	$h(n_4, k_2)$	$\rightarrow$	$\mathsf{decrypt}(n_4)$
SDecrypt:	$h(n_2, k_2), \mathsf{senc}(k_1, k_2)$	$\rightarrow$	k_1

Trusted Keys

Introduced in V2.20 of PKCS#11 (after Clulow's attacks)

SO can mark keys as trusted

Keys can be marked wrap_with_trusted

Can show security of this in our model

Full story in [Delaune, Kremer, Steel, CSF'08]

Summary

- Security APIs critical part of system design
- Have seen attacks on VSM, CCS, PKCS#11
- Formal analysis can find attaks, verify fixes
- More info at:

  http://www.lsv.ens-cachan.fr/~steel/

  (links to publications, security APIs FAQ, ASA Workshop, other researchers in API analysis, etc.)