Cryptographic device APIs: formal specification and verification

Graham Steel

INRIA & LSV, ENS de Cachan

Formal Methods and Tools for Security Workshop, 7 December 2011

Endpoint Security

> “Using encryption on the Internet is the equivalent of arranging an armoured car to deliver credit card information from someone living in a cardboard box to someone living on a park bench”

Prof Gene Spafford, Purdue University

> “the major risks to data on the Internet are at the endpoints - Trojans and rootkits on users’ computers, attacks against databases and servers, etc. and not in the network”

Bruce Schneier, CTO Security BT, Blog Jan 2009


Infineon
TPM


CAUTION
IBM
Lithium 3V
VARTA

Application Areas

Military: long history e.g. WW2 Enigma machines

Commercial:

Cash machines

- Encrypted PIN Pads (EPPs) and Hardware Security Modules (HSMs)

Cryptographic Smartcards

- used in SIM cards, credit cards, ID cards, transport...

Trusted Platform Module (TPM)

- now standard in most PC laptops

The future: Secure Elements in mobile phones, cars,...

Cryptographic Device APIs

Cryptographic Device APIs

Cryptographic Device APIs

Host machine

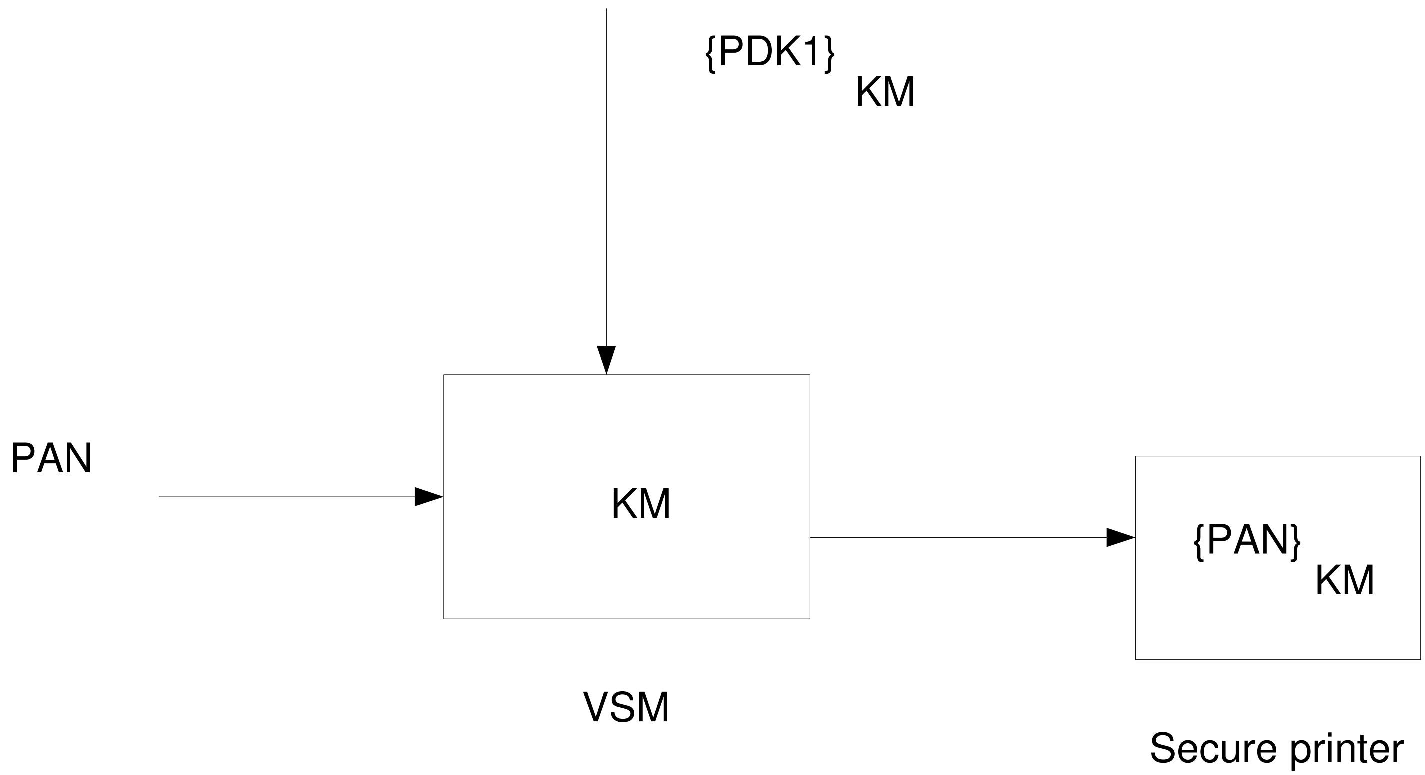
Trusted device

Security API

API attack example: VSM (Bond & Anderson, ’01)

Host machine	HSM
$\{ TMK1 \}_{KM}$	KM
$\{ PDK1 \}_{KM}$	

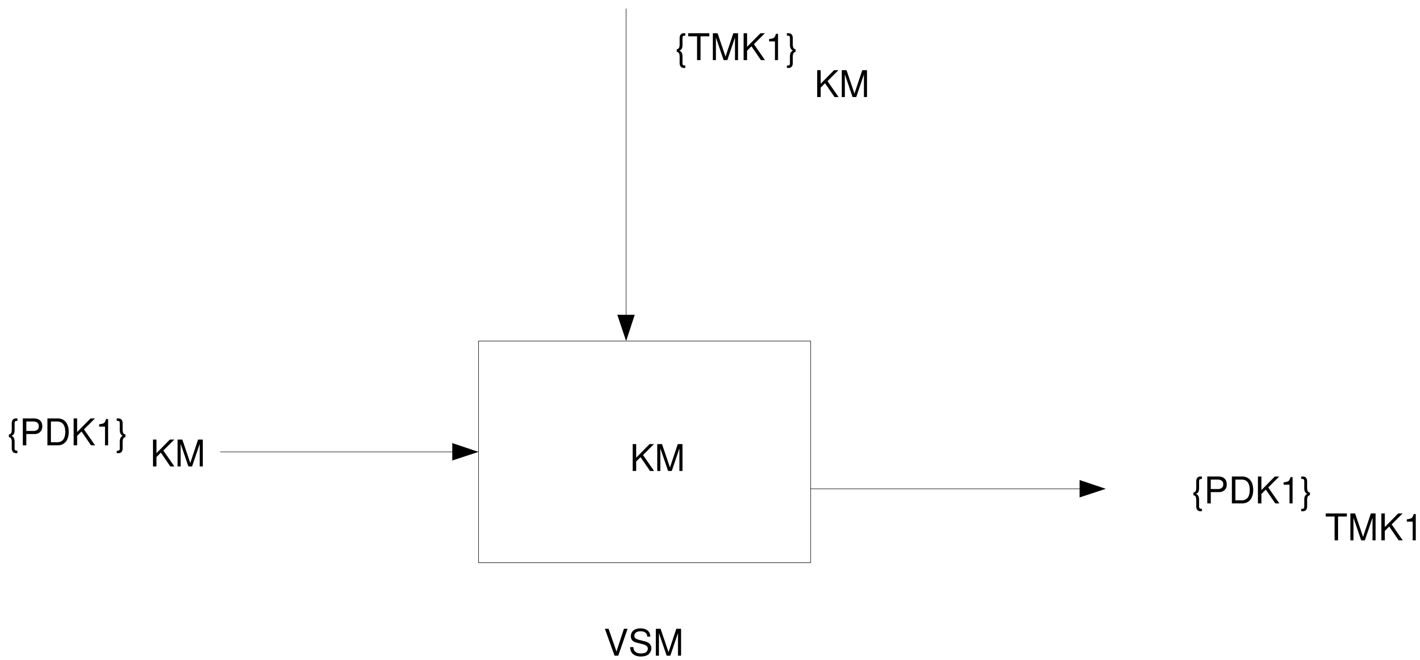
Example: Print Customer PIN



$$\text{Host} \longrightarrow \text{HSM} : \text{PAN}, \{ \text{PDK1} \}_{\text{KM}}$$

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

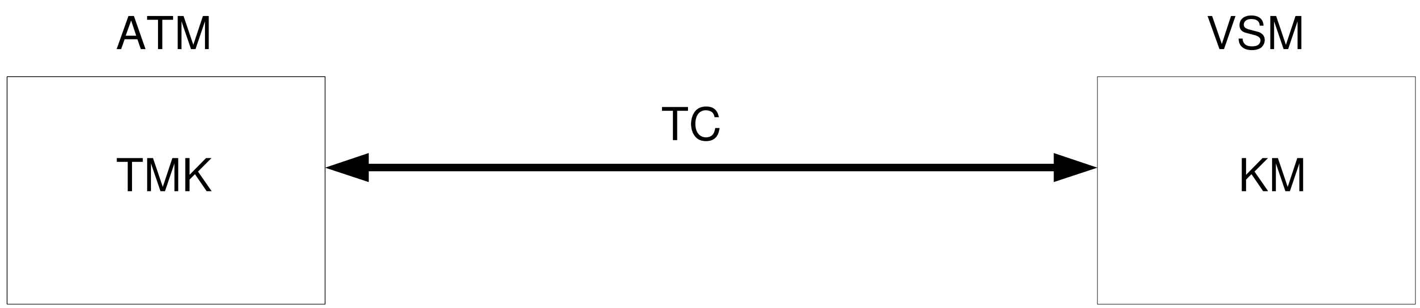
Example: Send PDK to Terminal



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

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

Terminal Comms Key



Managing Key Types

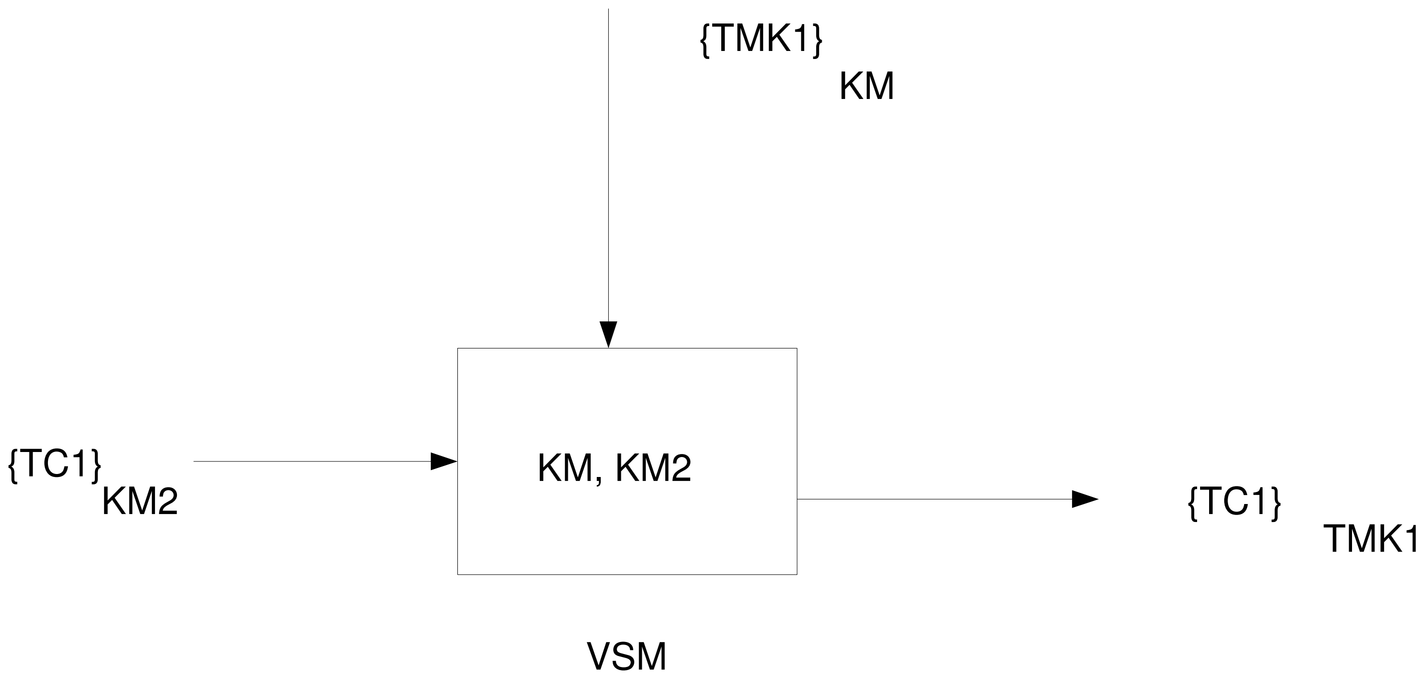
Host machine	VSM
$\{ TMK1 \}_{KM}$	KM
$\{ PDK1 \}_{KM}$	
	KM2
$\{ TC1 \}_{KM2}$	

Example: Enter TC key

$$\text{Host} \longrightarrow \text{HSM} : \text{TC}$$

$$\text{HSM} \longrightarrow \text{Host} : \{\, \text{TC} \,\}_{KM2}$$

Example: Send TC to Terminal



$$\text{Host} \longrightarrow \text{HSM} : \{ \text{TC} \}_{KM2}, \{ \text{TMK1} \}_{KM}$$

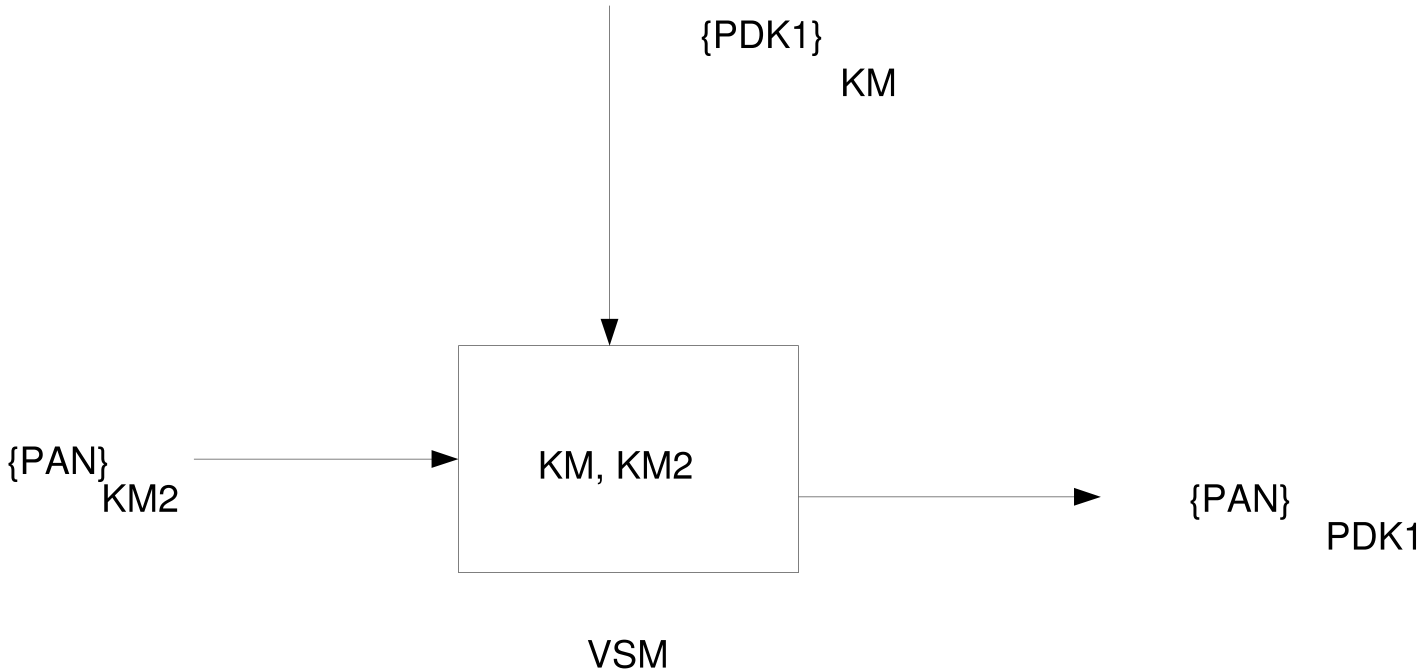
$$\text{HSM} \longrightarrow \text{Host} : \{ \text{TC} \}_{TMK1}$$

Attack - Step 1

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

$$\text{HSM} \longrightarrow \text{Spy} : \{\ \text{PAN}\ \}_{\text{KM2}}$$

Attack - Step 2



$$\text{Spy} \longrightarrow \text{HSM} : \{\text{PAN}\}_{\text{KM2}}, \{\text{PDK1}\}_{\text{KM}}$$

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

Dolev-Yao Modelling

(see talks passim)

Originally proposed for protocol analysis, can be adapted to APIs

Recap:

Bitstrings modelled as terms in an abstract algebra

Cryptographic algorithms modelled as functions on terms

Attacker controls network but cryptography is ‘perfect’

Model API security properties as secrecy, i.e. derivability of secret term

Secrecy in general undecidable (see e.g. [Durgin et al. ’99])

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

Semantics of model can be described by transition system

Analysing the model

Start with 'initial knowledge' of intruder

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

Apply API and intruder rules

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

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

e.g. VSM problem is `SWV237`, `SWV238` in (`www.tptp.org`)

Attack found in < 1 sec by several provers (Vampire, E,...), only E can find a model.

Implicit or explicit decryption?

$$\{x\}_y, y \quad \rightarrow \quad x$$

or

Explicit destructor $\mathrm{dec}(x, y)$

$$\mathrm{dec}(\{x\}_y, y) = x$$

There are equivalence results (Millen '03, Lynch '05), but these frequently do not apply to APIs ('EV freeness' conditions)

Authenticated encryption would seem to imply implicit decryption is sufficient (soundness results). But APIs only just starting to use this.

How to Model Fresh Data?

Choices include:

- Finite number of pre-identified values

  See e.g. the SATMC tool

- Truly fresh terms (using e.g. counter in global state or history of used terms)

  See e.g. inductive approach in Isabelle/HOL

- Abstractions using techniques from abstract interpretation

  See e.g. Proverif tool

What About Equational Theories?

e.g IBM CCA API

Encrypt data:

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

$$\begin{aligned} \mathsf{x}, \mathsf{y} &\longrightarrow \{\mathsf{x}\}_{\mathsf{y}} \\ \{\mathsf{x}\}_{\mathsf{y}}, \mathsf{y} &\longrightarrow \mathsf{x} \\ \mathsf{x}, \mathsf{y} &\longrightarrow \mathsf{x} \oplus \mathsf{y} \end{aligned}$$

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

What if the API has mutable state?

e.g. RSA PKCS#11 (see next talk)

Extend our form of rules:

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

Practical problem for tools: search for attack no longer monotonic, some state changes may exclude future steps

Further Reading

R. Focardi, F. L. Luccio, G. Steel. *An Introduction to Security API Analysis*. In FOSAD'VI, 2011.

V. Cortier, G. Keighren, G. Steel, *Automatic Analysis of the Security of XOR-based Key Management Schemes*, TACAS '07

S. Delaune, S. Kremer, G. Steel, *Formal Analysis of PKCS#11*, CSF 2008

D. Longley and S. Rigby, *An Automatic Search for Security Flaws in Key Management Schemes*, Computers and Security, 1992

M. Bond and R. Anderson, *API Level Attacks on Embedded Systems*, IEEE Computer Magazine, 2001

Dagstuhl Seminar "Analysis of Security APIs" (November 2012) and workshop series ASA (next one in Boston, June 2012)