Attacking and Fixing PKCS#11 Security Tokens

Matteo Bortolozzo, Matteo Centenaro
Riccardo Focardi and Graham Steel

Università Ca' Foscari, Venezia
and
LSV, INRIA & CNRS & ENS-Cachan

RSA PKCS#11

Describes ‘cryptoki’: cryptographic token interface

Widely adopted in industry for authentication tokens, smartcards (and HSMs, other devices, . . .)

RSA PKCS#11

Describes ‘cryptoki’: cryptographic token interface

Widely adopted in industry for authentication tokens, smartcards (and HSMs, other devices, . . .)

Authentication tokens used for secure login to VPN etc.

Devices cost from 20 to 400 USD, global market estimated at 5 billion USD by InfoSecurity Magazine

RSA PKCS#11

Describes ‘cryptoki’: cryptographic token interface

Widely adopted in industry for authentication tokens, smartcards (and HSMs, other devices, . . .)

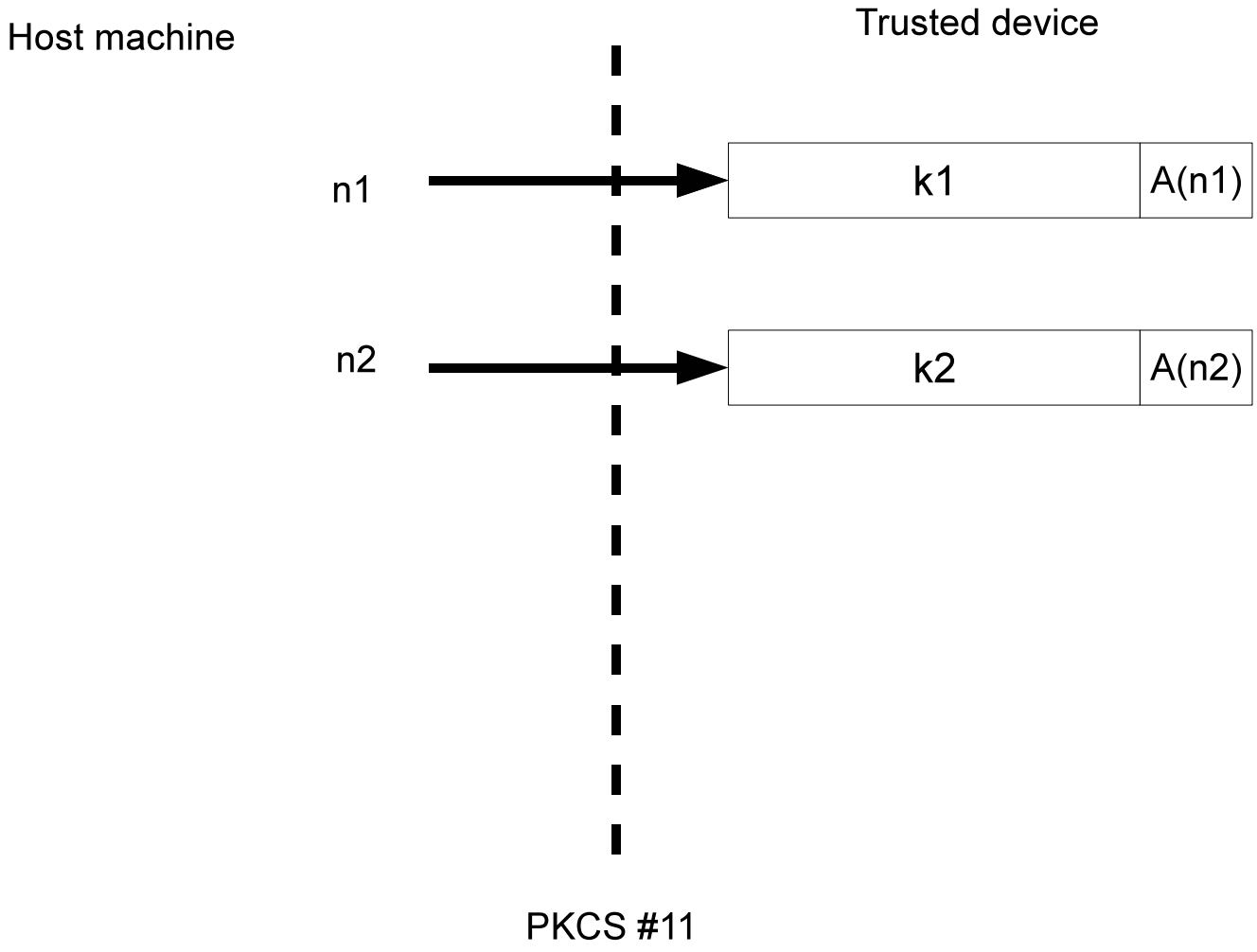
Authentication tokens used for secure login to VPN etc.

Devices cost from 20 to 400 USD, global market estimated at 5 billion USD by InfoSecurity Magazine

Cryptoki provides a logical view of objects on the token

Keys (etc.) stored on the device and accessed by *handles*

Attributes stored with keys to control usage


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

PKCS#11 Security

Section 7 of standard:

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.

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”

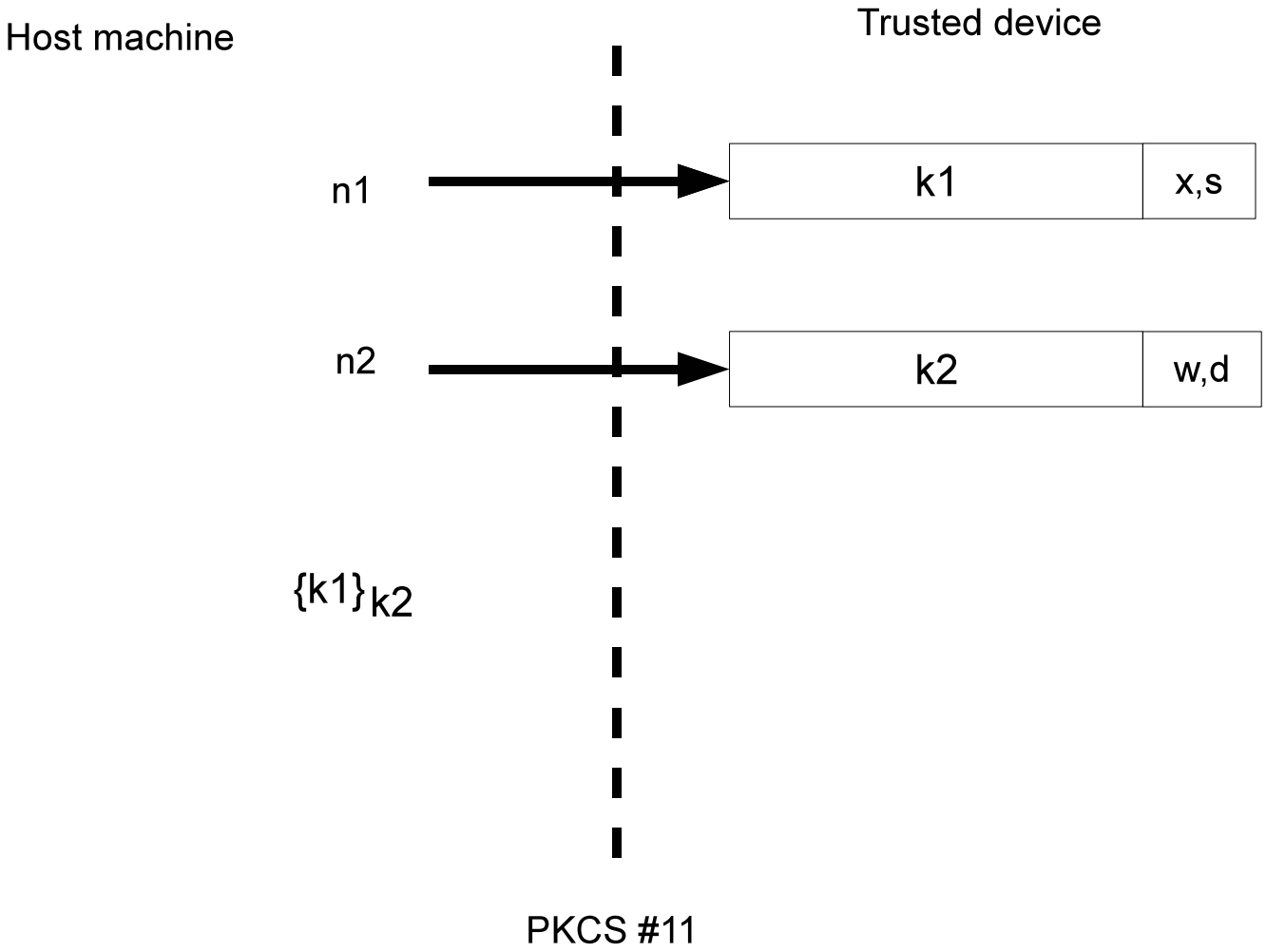
PKCS#11 Security

Section 7 of standard:

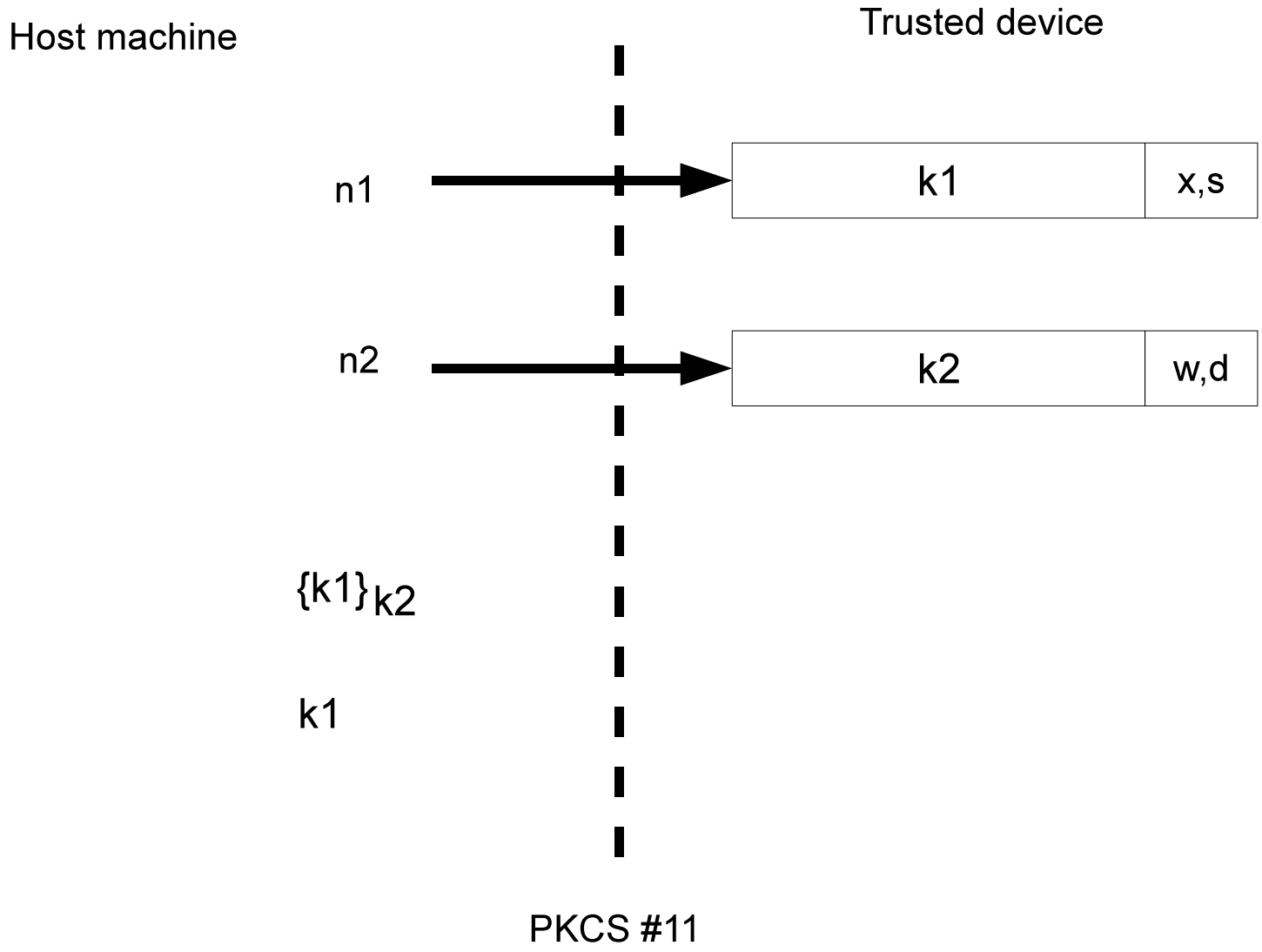
“1. Access to private objects on the token, and possibly to cryptographic functions and/or certificates on the token as well, requires a PIN.

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,d
{k1}k2
PKCS #11

Clulow, CHES 2003



Formal Model (Delaune, Kremer, S., CSF 2008)

Abstract 'Dolev-Yao' style

$\mathsf{h}(\mathsf{n1}, \mathsf{k1})$ - a handle $\mathsf{n1}$ for key $\mathsf{k1}$ (h is a *private symbol*)

$\mathsf{a1}(\mathsf{n1})$ - setting of attribute $\mathsf{a1}$ for handle $\mathsf{n1}$

Command :

$$\mathsf{input}; \mathsf{state} \xrightarrow{\mathsf{new}} \mathsf{output}; \mathsf{state}'$$

Key Management - 1

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

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

Key Management - 2

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

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

Key Management - 3

Wrap :

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

Unwrap :

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

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

Key Usage

Encrypt :

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

Decrypt :

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

Fix decrypt/wrap, (and encrypt/unwrap):

Fix decrypt/wrap, (and encrypt/unwrap):

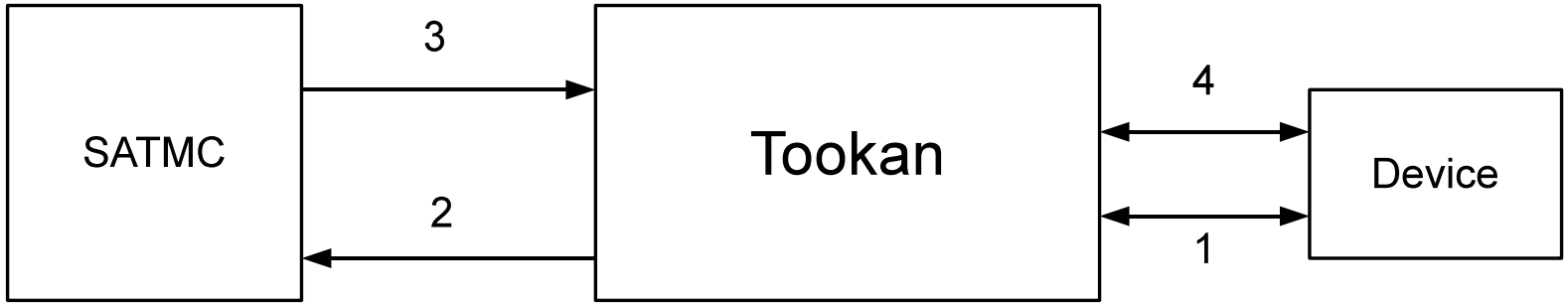
Intruder knows: $h(n_1, k_1)$, $h(n_2, k_2)$, k_3

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

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

TOOKAN

‘Tool for cryptoKi Analysis’



Templates

$$\text{KeyGenerate}: \quad \xrightarrow{\text{new n,k}} \quad \mathsf{h}(\mathsf{n},\mathsf{k}); \mathcal{A}(\mathsf{n},\mathsf{B}) \qquad (\text{with } \mathsf{B} \in \mathcal{G})$$

$$\text{KeyPairGenerate}: \quad \xrightarrow{\text{new n,s}} \quad \mathsf{h}(\mathsf{n},\mathsf{s}), \mathsf{pub}(\mathsf{s}); \mathcal{A}(\mathsf{n},\mathsf{B}) \quad (\text{with } \mathsf{B} \in \mathcal{G})$$

$$\text{Unwrap}\,(\text{sym}/\text{sym}):$$

$$\mathsf{h}(\mathsf{x},\mathsf{y}_2), \{\!|\mathsf{y}_1|\!\}_{\mathsf{y}_2}; \mathsf{unwrap}(\mathsf{x},\top) \quad \xrightarrow{\text{new n}_1} \quad \mathsf{h}(\mathsf{n}_1,\mathsf{y}_1); \mathcal{A}(\mathsf{n}_1,\mathsf{B}) \quad (\text{with } \mathsf{B} \in \mathcal{U})$$

$$\text{CreateObject}: \quad \mathsf{x}; \quad \xrightarrow{\text{new n}} \quad \mathsf{h}(\mathsf{n},\mathsf{x}); \mathcal{A}(\mathsf{n},\mathsf{B}) \quad (\text{with } \mathsf{B} \in \mathcal{C})$$

Configuration Language

Functions

Attributes

Always on/off

Conflicts

Tied

Templates

Flags

(see http://secgroup.ext.dsi.unive.it/tookan for full description)

Abstractions for Proof (based on Fröschle & Steel WITS '09)

$$\mathsf{KeyGenerate}: \quad \rightarrow \quad \mathsf{h}(\mathsf{n_i}, \mathsf{k_i}); \mathcal{A}(\mathsf{n_i}, \mathsf{B_i}) \qquad (\text{with } \mathsf{B_i} \in \mathcal{G})$$

$$\mathsf{KeyPairGenerate}: \quad \rightarrow \quad \mathsf{h}(\mathsf{n_j}, \mathsf{s_j}), \mathsf{pub}(\mathsf{s_j}); \mathcal{A}(\mathsf{n_j}, \mathsf{B_j}) \quad (\text{with } \mathsf{B_j} \in \mathcal{G})$$

$\mathsf{Unwrap}\,(\mathsf{sym}/\mathsf{sym})$:

$$\mathsf{h}(\mathsf{x}, \mathsf{y_2}), \{\!|\mathsf{y_1}|\!\}_{\mathsf{y_2}}; \mathsf{unwrap}(\mathsf{x}, \top) \quad \rightarrow \quad \mathsf{h}(\mathsf{n_k}, \mathsf{y_1}); \mathcal{A}(\mathsf{n_k}, \mathsf{B_k}) \quad (\text{with } \mathsf{B_k} \in \mathcal{U})$$

$$\mathsf{CreateObject}: \quad \mathsf{x}; \quad \rightarrow \quad \mathsf{h}(\mathsf{n_l}, \mathsf{x}); \mathcal{A}(\mathsf{n_l}, \mathsf{B_l}) \quad (\text{with } \mathsf{B_l} \in \mathcal{C})$$


TOOKAN

Device		Supported Functionality						Attacks found				
Brand	Model	s	as	cobj	chan	w	ws	wd	rs	ru	su	Tookan
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 7 received notification at least 5 months before publication.

We offered to publish responses on project website

Manufacturer Reaction

All 7 received notification at least 5 months before publication.

We offered to publish responses on project website

RSA sent response, registered vulnerability with Mitre (CVE-2010-3321), will issue patch details today

Aladdin (now Safenet) sent a 2-page response for website

Manufacturer Reaction

All 7 received notification at least 5 months before publication.

We offered to publish responses on project website

RSA sent response, registered vulnerability with Mitre (CVE-2010-3321), will issue patch details today

Aladdin (now Safenet) sent a 2-page response for website

Bull invited me for a private meeting at their HQ

Manufacturer Reaction

All 7 received notification at least 5 months before publication.

We offered to publish responses on project website

RSA sent response, registered vulnerability with Mitre (CVE-2010-3321), will issue patch details today

Aladdin (now Safenet) sent a 2-page response for website

Bull invited me for a private meeting at their HQ

Gemalto responded to Cyberflex vulnerability, but not to SafeSite, and not to request to publish their reponse.

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

OpencryptokiX

IBM Opencryptoki is a library including a software token

Vulnerable to many attacks (but it’s a software token)

OpencryptokiX

IBM Opencryptoki is a library including a software token

Vulnerable to many attacks (but it's a software token)

We have coded two fixed versions

- one implements config from Fröschle & Steel WITS '09
- one is a new fix with no new crypto mechanisms

  Uses a carefully chosen set of templates $\mathcal{G} = \{wu, ed\}, \mathcal{U} = \{eu\}$

OpencryptokiX

IBM Opencryptoki is a library including a software token

Vulnerable to many attacks (but it's a software token)

We have coded two fixed versions

- one implements config from Fröschle & Steel WITS '09
- one is a new fix with no new crypto mechanisms

  Uses a carefully chosen set of templates $\mathcal{G} = \{wu, ed\}, \mathcal{U} = \{eu\}$

Available to download from

http://secgroup.ext.dsi.unive.it/cryptokix

Conclusions

Tookan: our tool for formal analysis of PKCS#11 configurations

OpencryptokiX: a sandbox for trying token configurations

Bees: a library for programming PKCS#11 tokens using symbolic model language

Conclusions

Tookan: our tool for formal analysis of PKCS#11 configurations

OpencryptokiX: a sandbox for trying token configurations

Bees: a library for programming PKCS#11 tokens using symbolic model language

State of art of tokens not great (10/18 vulnerable, the rest very limited functionality)

Some manufacturers patching, no reaction from others

Conclusions

Tookan: our tool for formal analysis of PKCS#11 configurations

OpencryptokiX: a sandbox for trying token configurations

Bees: a library for programming PKCS#11 tokens using symbolic model language

State of art of tokens not great (10/18 vulnerable, the rest very limited functionality)

Some manufacturers patching, no reaction from others

Maybe we need a new standard with modern crypto? (OASIS, IEEE SISWG,. . .)

Conclusions

Tookan: our tool for formal analysis of PKCS#11 configurations

OpencryptokiX: a sandbox for trying token configurations

Bees: a library for programming PKCS#11 tokens using symbolic model language

State of art of tokens not great (10/18 vulnerable, the rest very limited functionality)

Some manufacturers patching, no reaction from others

Maybe we need a new standard with modern crypto? (OASIS, IEEE SISWG,. . .)

More details in the paper or online:

http://secgroup.ext.dsi.unive.it/tookan