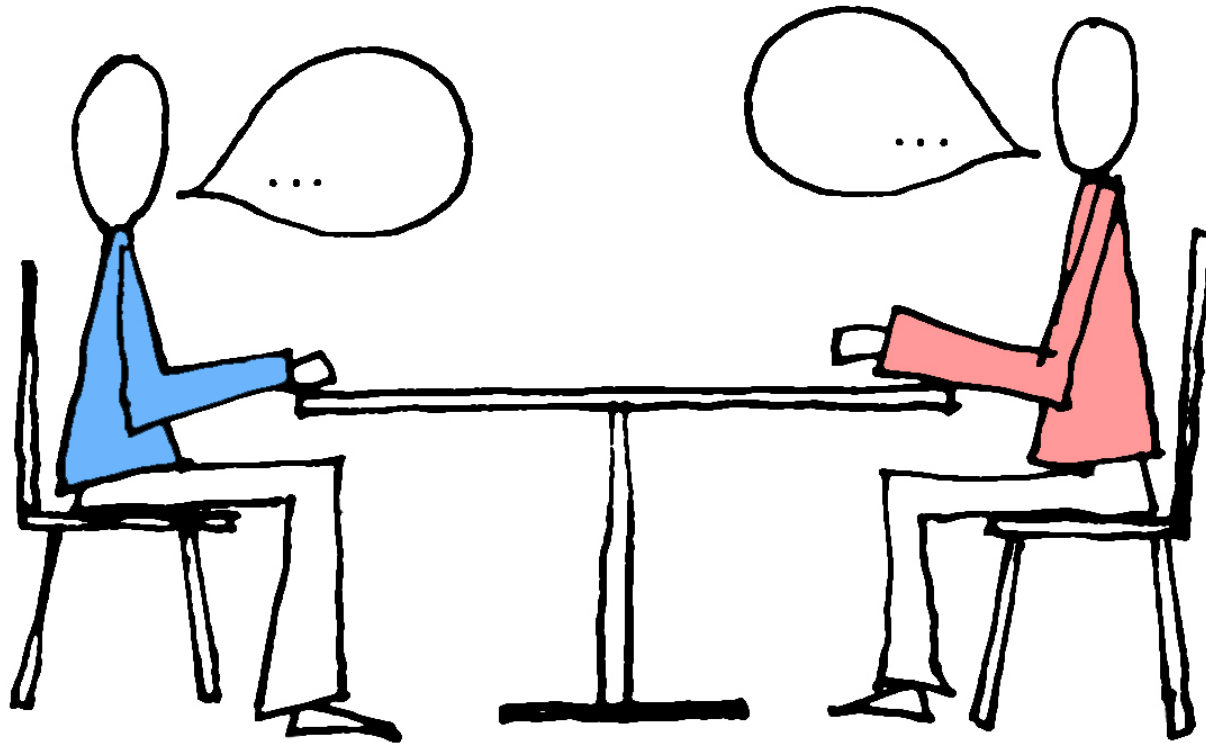


# Analysis of Cryptographic APIs

Graham Steel

LSV, INRIA & CNRS & ENS-Cachan

# Cryptography in Practice v1



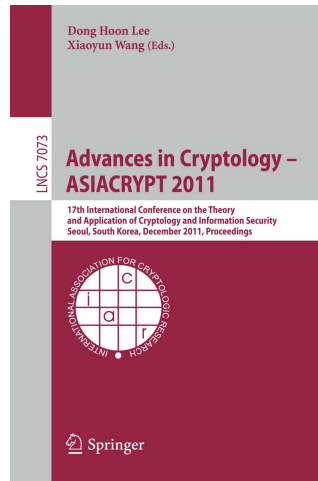
PM talks to client to understand security goals and threats

# Cryptography in Practice v1



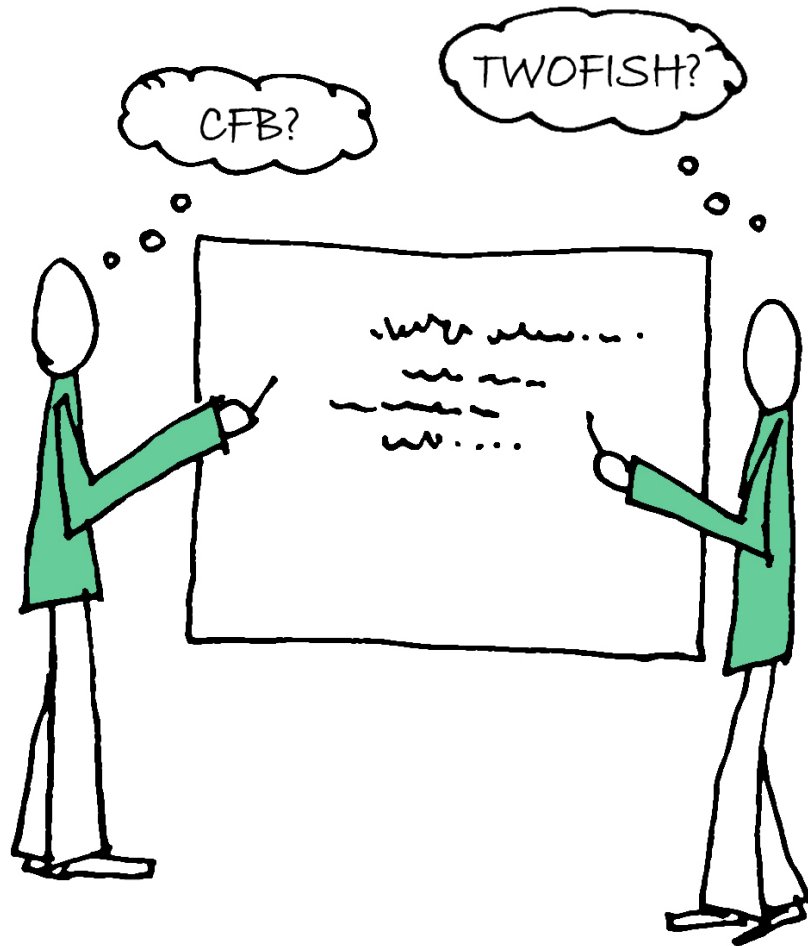
Engineer and PM discuss, decide formal requirements of crypto schemes

# Cryptography in Practice v1



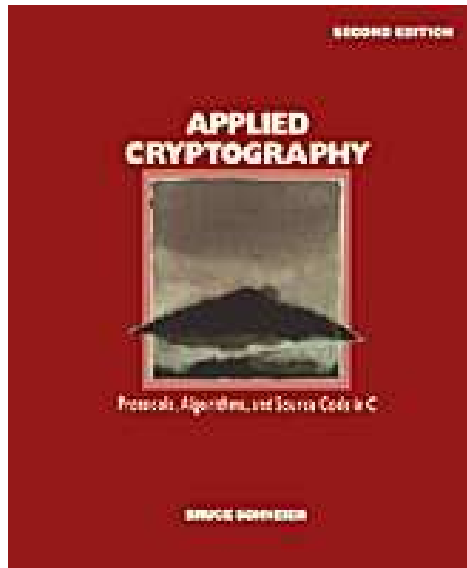
Engineer consults relevant literature for provably secure scheme

# Cryptography in Practice v2



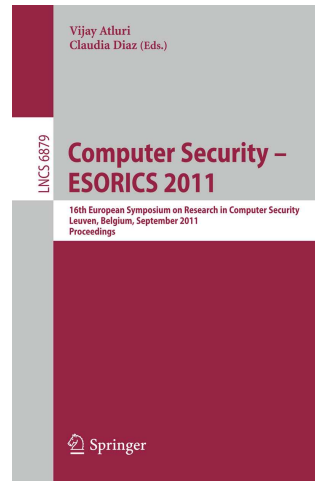
Engineers get together and plan some fun crypto stuff

# Cryptography in Practice v2



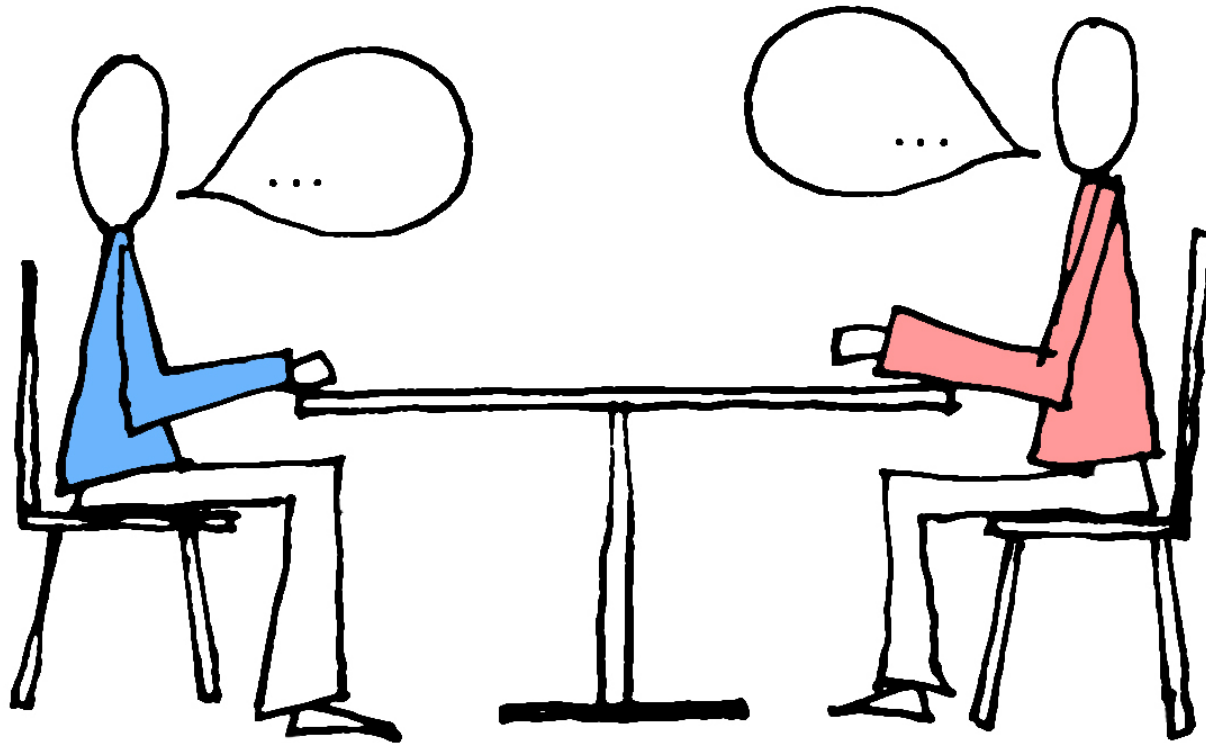
Literature consulted

# Cryptography in Practice v2



Literature augmented by attacks on the system

## Cryptography in Practice v3



PM instructed by client: "Solution must use PKCS#11 hardware"

# Cryptography in Practice v3



PM instructs engineer: "Use PKCS#11 API"

# Cryptography in Practice v3

?

# RSA Public Key Cryptography Standard (PKCS) 11

PKCS #1 describes the RSA encryption algorithm, padding etc.

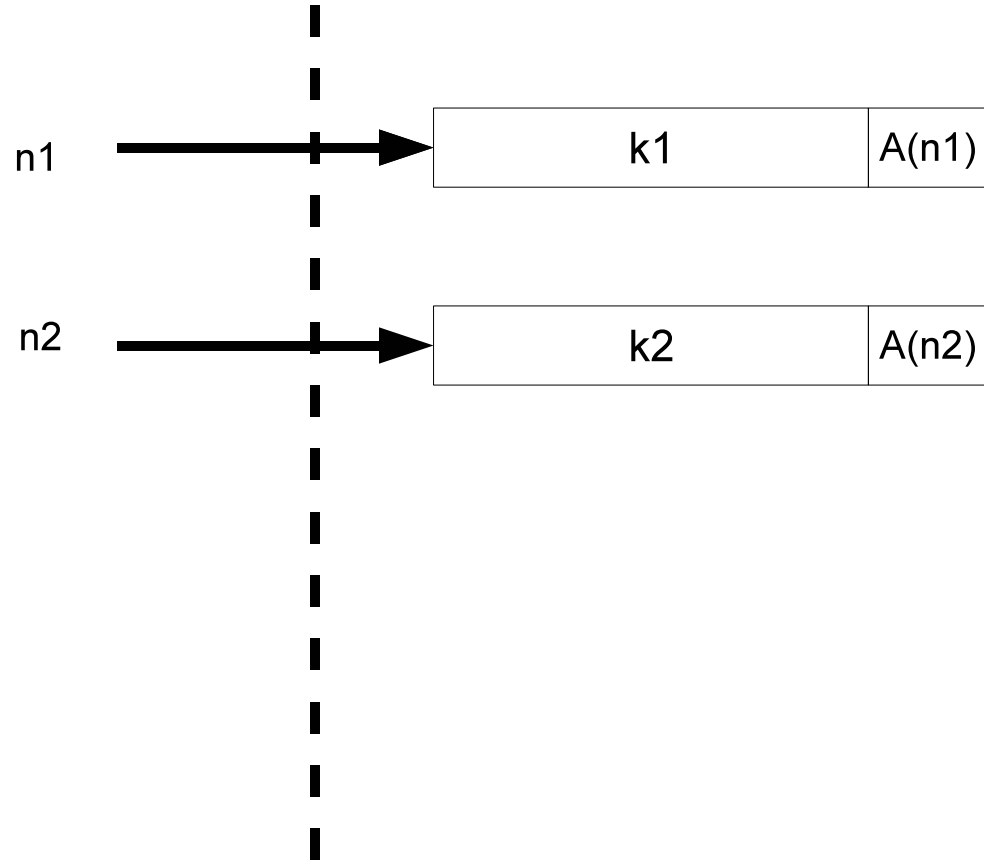
PKCS#11 Describes 'cryptoki': cryptographic token interface

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

Version 1.0 of PKCS#11 1995, current version 2.20 2004

Host machine

Trusted device



PKCS #11

## Generating keys with PKCS#11

A *key template* is a partial specification of *key attributes*

Templates are used for creating, manipulating, and searching for objects

## Generating keys with PKCS#11

A *key template* is a partial specification of *key attributes*

Templates are used for creating, manipulating, and searching for objects

C\_GenerateKey :

$$\mathcal{T} \xrightarrow{\text{new } n, k} h(n, k); \mathcal{T}$$

## Setting Key Attributes

C\_SetAttributeValue :

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

$\mathcal{T}$  can specify new values for any attributes, but may cause  
CKR\_TEMPLATE\_INCONSISTENT, CKR\_ATTRIBUTE\_READ\_ONLY

## Wrap and Unwrap

Wrap :

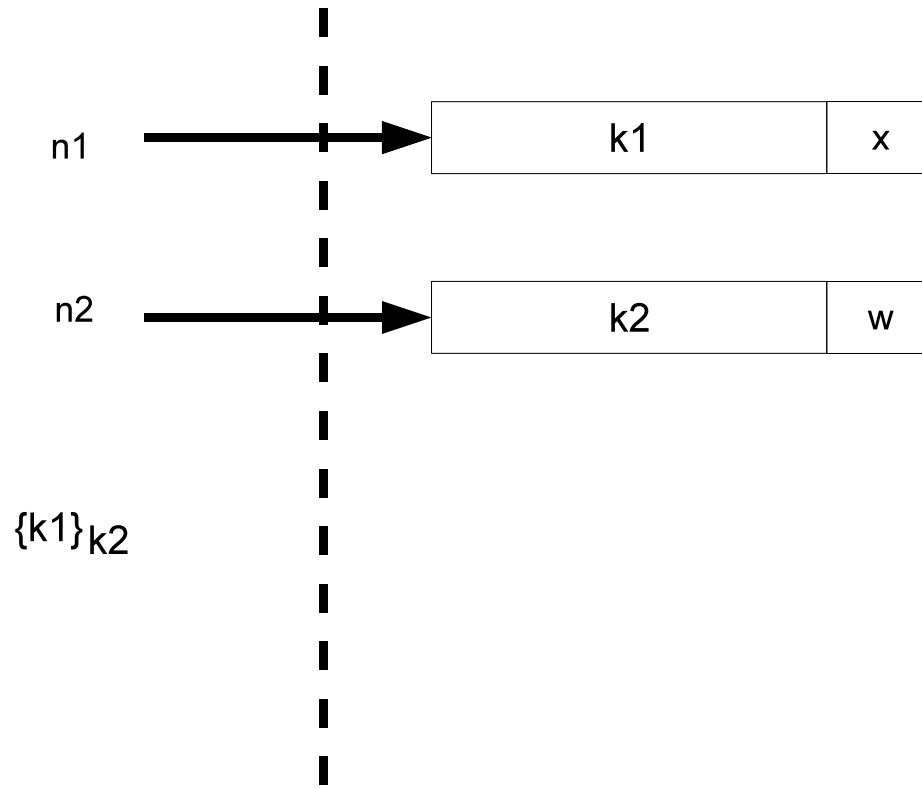
$$h(x_1, y_1), h(x_2, y_2); \text{wrap}(x_1), \quad \rightarrow \quad \{y_2\}_{y_1} \\ \text{extract}(x_2)$$

Unwrap :

$$h(x_2, y_2), \{y_1\}_{y_2}, \mathcal{T}; \text{unwrap}(x_2) \xrightarrow{\text{new } n_1} h(n_1, y_1); \text{extract}(n_1), \mathcal{T}$$

Host machine

Trusted device



## Key Usage

Encrypt :

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

Decrypt :

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

# 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

$\{k1\}_{k2}$

PKCS #11

Host machine

Trusted device

n1



k1

x,s

n2



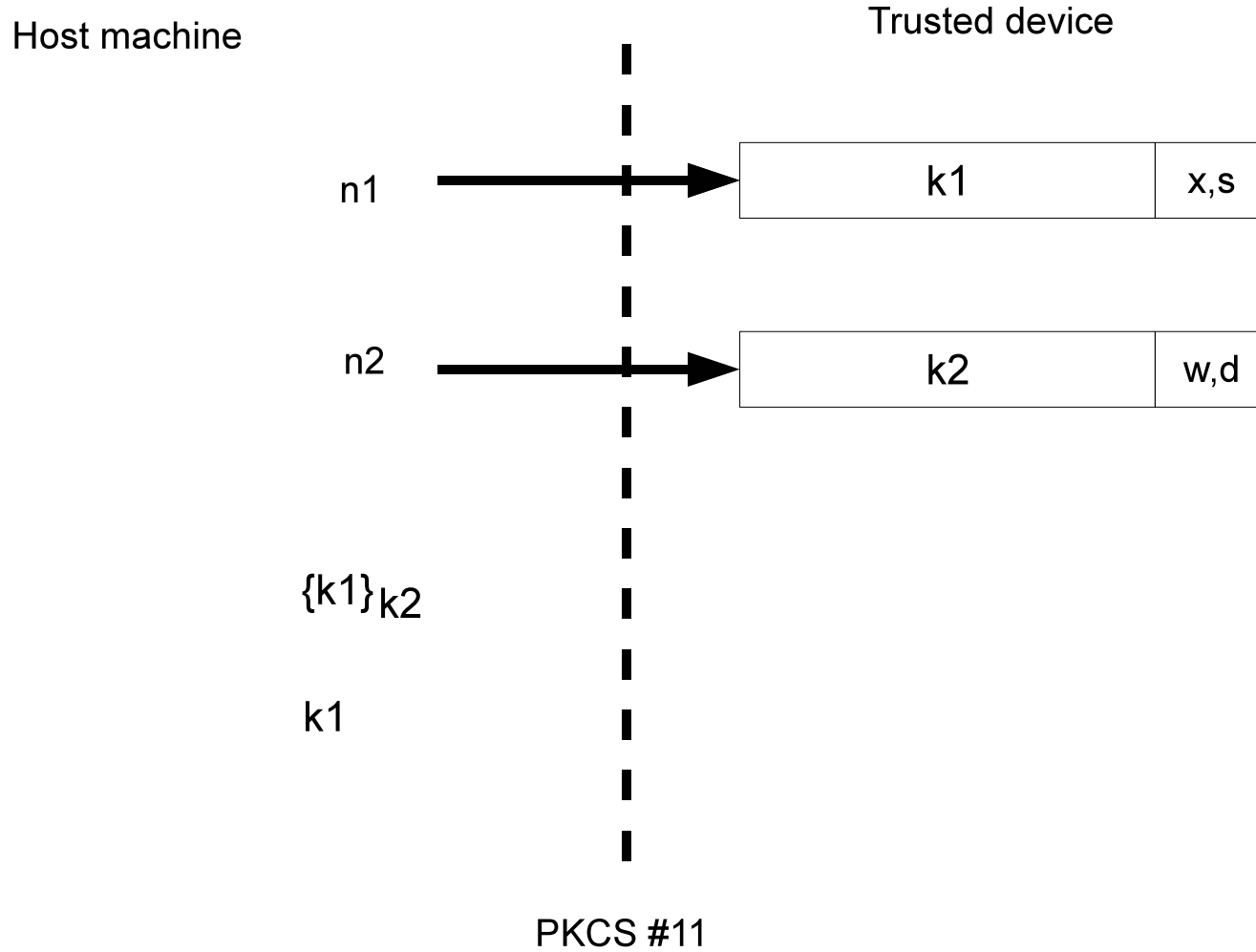
k2

w,d

{k1}k2

PKCS #11

# Clulow, CHES 2003



**Prevent a key from doing decrypt and wrap..**

## Prevent a key from doing decrypt and wrap..

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

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

Set\_wrap:  $h(n_2, k_2) \rightarrow ;\text{wrap}(n_2)$

Set\_wrap:  $h(n_1, k_1) \rightarrow ;\text{wrap}(n_1)$

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

Set\_unwrap:  $h(n_1, k_1) \rightarrow ;\text{unwrap}(n_1)$

Unwrap:  $h(n_1, k_1), \{k_2\}_{k_1} \xrightarrow{\text{new } n_3} h(n_3, k_2)$

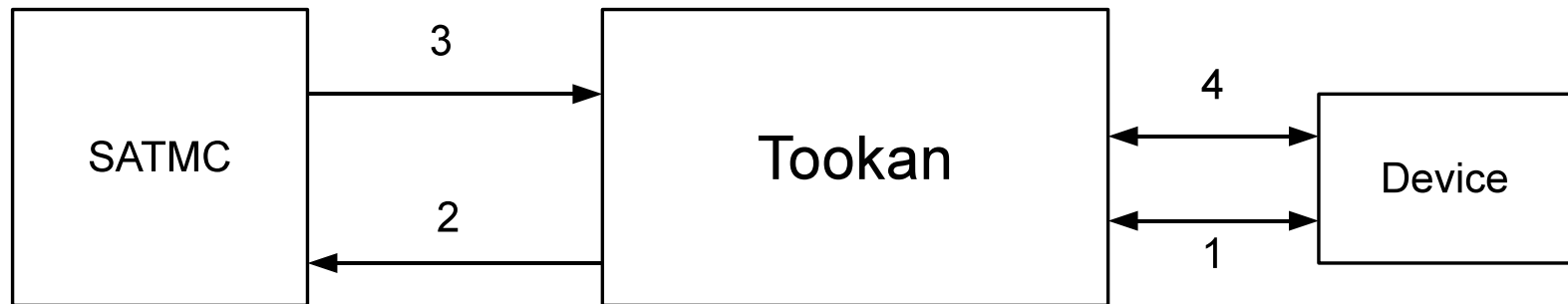
Wrap:  $h(n_2, k_2), h(n_1, k_1) \rightarrow \{k_1\}_{k_2}$

Set\_decrypt:  $h(n_3, k_2) \rightarrow ;\text{decrypt}(n_3)$

Decrypt:  $h(n_3, k_2), \{k_1\}_{k_2} \rightarrow k_1$

# TOOKAN

‘Tool for cryptoKi Analysis’



[Bortolozzo, Centenaro, Focardi & S., CCS 2010]

## Choice of Algorithms

PKCS#11 includes many *mechanisms*, some rather old (e.g. single DES)

This can lead to problems when e.g. a weak mechanism is used to encrypt a key for a strong one

## Choice of Algorithms

PKCS#11 includes many *mechanisms*, some rather old (e.g. single DES)

This can lead to problems when e.g. a weak mechanism is used to encrypt a key for a strong one

Only recent draft v2.30 proposes support for authenticated encryption modes (GCM, CCM)

For asymmetric crypto, OAEP is supported by PKCS#1 v1.5 still in standard and widely used

Unwrap command is problematic with these modes



| Device  |                  | Supported Functionality |    |      |      |   |    | Attacks found |    |    |    | Tookan |    |
|---------|------------------|-------------------------|----|------|------|---|----|---------------|----|----|----|--------|----|
| Brand   | Model            | s                       | as | cobj | chan | w | ws | wd            | rs | ru | su |        |    |
| 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),  
issued security advisory 6 Oct 2010

Aladdin (now Safenet) and Gemalto sent a 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), 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)

## 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), 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 by Boeing and a major UK-based bank.

## Testing on HSMs

- Recently, Tookan has been deployed in testing PKCS#11-compatible HSMs
- HSMs typically have a more sophisticated attribute policy, and support many configuration options
- Still Tookan finds many attacks
- Highly diverse implementations of padding checks
- Working on Tookan v.2 with an improved, more general reverse engineering algorithm

## **Summary and Conclusions**

Demoed Tookan: an effective tool for formal analysis of PKCS#11 configurations

## Summary and Conclusions

Demoed Tookan: an effective tool for formal analysis of PKCS#11 configurations

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

Recently: testing on HSMs. Interesting results.

PKCS#11 is clearly a difficult standard to use.

## Summary and Conclusions

Demoed Tookan: an effective tool for formal analysis of PKCS#11 configurations

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

Recently: testing on HSMs. Interesting results.

PKCS#11 is clearly a difficult standard to use.

Not yet clear whether emerging API standards (KMIP, IEEE 1619..) will be better.

## Summary and Conclusions

Demoed Tookan: an effective tool for formal analysis of PKCS#11 configurations

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

Recently: testing on HSMs. Interesting results.

PKCS#11 is clearly a difficult standard to use.

Not yet clear whether emerging API standards (KMIP, IEEE 1619..) will be better.

Project webpage:

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