

# PKCS#11: Secure Configurations and New Attacks

Sibylle Fröschle<sup>1</sup>   Nils Sommer<sup>2</sup>

<sup>1</sup>University of Oldenburg  
Germany

<sup>2</sup>MWR InfoSecurity  
UK

June 30, 2011

# Public Key Cryptographic Standard (PKCS) #11



Smartcard



eToken



Hardware Security Module



## Cryptoki

Encrypt Data  
Decrypt Data  
Generate Key  
Export Key  
Import Key  
...



Alice

# Public Key Cryptographic Standard (PKCS) #11



Smartcard



eToken



Hardware Security Module



## Cryptoki

Encrypt Data  
Decrypt Data  
Generate Key  
Export Key  
Import Key  
...



# Cryptoki

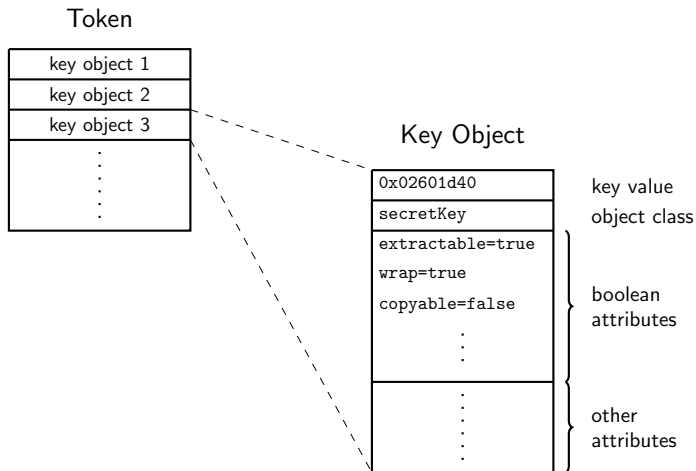
- ▶ Logical view:

Token

key object 1
key object 2
key object 3
⋮

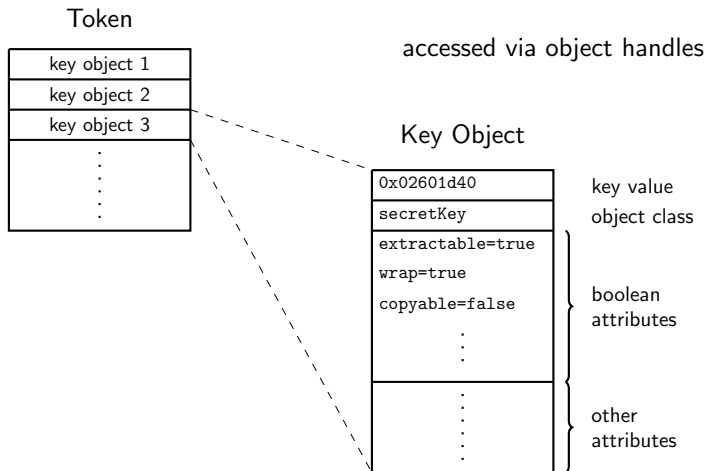
# Cryptoki

► Logical view:



# Cryptoki

► Logical view:

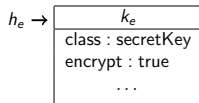


# Cryptoki Commands

- ▶ Encrypt data  $d$  under key  $k_e$

H  $\rightarrow$  T: Encrypt  $h_e$   $d$

T  $\rightarrow$  H:  $\{d\}_{k_e}$

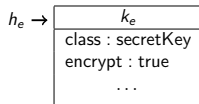


# Cryptoki Commands

- ▶ Encrypt data  $d$  under key  $k_e$

H  $\rightarrow$  T: `Encrypt  $h_e$   $d$`

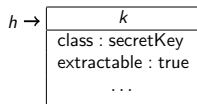
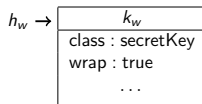
T  $\rightarrow$  H: `{ $d$ } $_{k_e}$`



- ▶ Export key  $k$  wrapped under  $k_w$

H  $\rightarrow$  T: `WrapKey  $h_w$   $h$`

T  $\rightarrow$  H: `{ $k$ } $_{k_w}$`

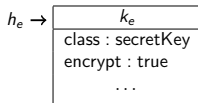


# Cryptoki Commands

- ▶ Encrypt data  $d$  under key  $k_e$

H  $\rightarrow$  T: Encrypt  $h_e$   $d$

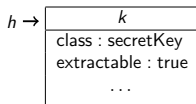
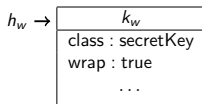
T  $\rightarrow$  H:  $\{d\}_{k_e}$



- ▶ Export key  $k$  wrapped under  $k_w$

H  $\rightarrow$  T: WrapKey  $h_w$   $h$

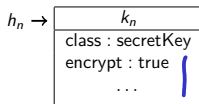
T  $\rightarrow$  H:  $\{k\}_{k_w}$



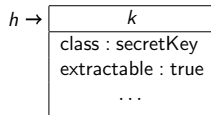
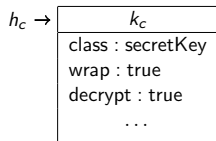
- ▶ Generate fresh symmetric key with attributes as specified by template

H  $\rightarrow$  T: GenerateKey (encrypt, true) ...

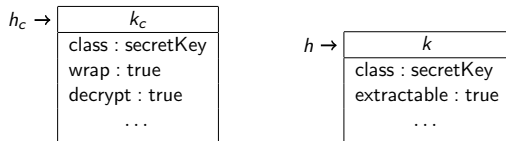
T  $\rightarrow$  H:  $h$ , handle to new object



# Key-Separation-Angriff (Clulow, 2003)



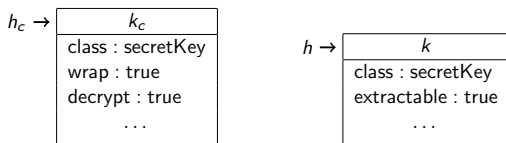
# Key-Separation-Angriff (Clulow, 2003)



H  $\rightarrow$  T : `WrapKey`  $h_c$   $h$

T  $\rightarrow$  H :  $\{k\}_{k_c}$

# Key-Separation-Angriff (Clulow, 2003)



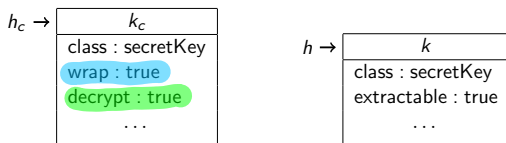
H  $\rightarrow$  T : WrapKey  $h_c$   $h$

T  $\rightarrow$  H :  $\{k\}_{k_c}$

H  $\rightarrow$  T : Decrypt  $h_c$   $\{k\}_{k_c}$

T  $\rightarrow$  H :  $k$

# Key-Separation-Angriff (Clulow, 2003)



$H \rightarrow T : \text{WrapKey } h_c \ h$   
 $T \rightarrow H : \{k\}_{k_c}$

$H \rightarrow T : \text{Decrypt } h_c \ \{k\}_{k_c}$   
 $T \rightarrow H : k$

If the USE of key  $k_c$  includes **wrapping** it shouldn't have ROLE **data decryption!**

# PKCS#11-Security

- ▶ Many other logical attacks! (Delaune et al. CSF'08)
- ▶ The attacks are real!

As shown by experiments on real PKCS#11 tokens:

- ▶ Bortolozzo et al., ACM CCS'10
- ▶ Sommer, Master thesis'09, Fröschle and Sommer, FAST'10



Fortunately ...

# Provably Secure Configurations

## We already know from ASA'10 ...

Keys that are generated with `extractable = false` are secure!  
(against any logical attacks)

- ▶ Aladdin eToken
- ▶ Siemens HiPath Scurity
- ▶ Estonian ID



The pre-configured private key of each of these tokens is secure!

# What if you need to export a sensitive key?

Use Case ATM Key Management:

- ▶ set up a 'key of the day' to encrypt transactions to be sent from ATM 1 to Bank A

# What if you need to export a sensitive key?

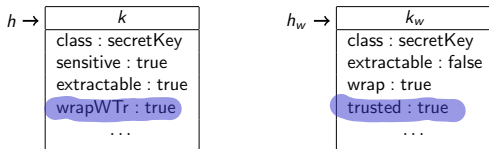
Use Case ATM Key Management:

- ▶ set up a 'key of the day' to encrypt transactions to be sent from ATM 1 to Bank A

Cryptoki offers the following features:

- ▶ Wrap with Trusted
- ▶ Wrap and Unwrap Templates

# Wrap with Trusted



1. If a key object has `wrapWTr = true` then it can only be wrapped by key objects with `trusted = true`.

H  $\rightarrow$  T: WrapKey  $h_w$   $h$   
T  $\rightarrow$  H:  $\{k\}_{k_w}$

2. Attribute `trusted` can only be set by the Security Officer.

# Security Goal

wrap-with-trusted = 

wrapWTr : true
sensitive : true
...

## Origin-Secrecy of wrap-with-trusted:

Every key  $k$  that is generated on the token with type wrap-with-trusted will always remain unknown to the intruder.

## How to achieve it?

trusted = 

trusted : true ...
-----------------------

1. How to set-up keys of type trusted?
2. Additional constraints on the functionality necessary?

# How to achieve it?

trusted = 

trusted : true ...
-----------------------

1. How to set-up keys of type trusted?
2. Additional constraints on the functionality necessary?

Analyse by identifying the potential attacks ...

(Here only symmetric keys but pk analogous.)

# Attack 1: Wrap with Known Trusted Key

$h_w \rightarrow$

$k_w$
wrap : true
trusted : true
...

$h \rightarrow$

$k$
sensitive : true
wrapWTr : true
extractable : true
...

target

H  $\rightarrow$  T: WrapKey  $h_w h$

T  $\rightarrow$  H:  $\{k\}_{k_w}$

$I$  obtains or has already obtained  $k_w$ .

H  $\rightarrow$  H: IDecrypt  $k_w \{k\}_{k_w}$

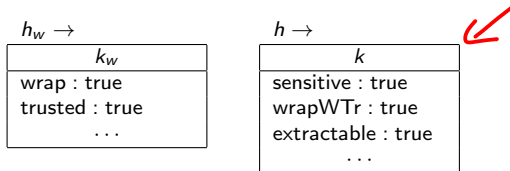
H  $\rightarrow$  H:  $k$

# Security Subgoal 1

## Secrecy of *trusted*

Every key  $k$  that is available on the token with type *trusted* has always been and will always remain unknown to the intruder.

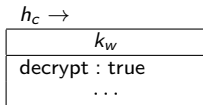
## Attack 2: Wrap/Decrypt with Trusted Key



H  $\rightarrow$  T: WrapKey  $h_w h$

T  $\rightarrow$  H:  $\{k\}_{k_w}$

I obtains or has already obtained:



H  $\rightarrow$  T: Decrypt  $h_c \{k\}_{k_w}$

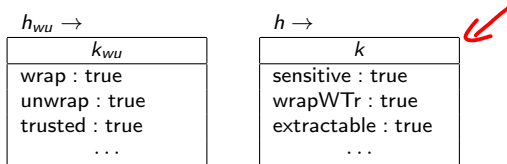
T  $\rightarrow$  H:  $k$

## Security Subgoal 2

### *Absence of role `decrypt` for `trusted`*

Every key  $k$  that is available with type *trusted* has never been and will never be enabled for role *decrypt*.

## Attack 3: Downgrade *sensitive*

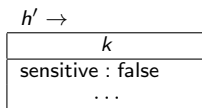


H  $\rightarrow$  T: WrapKey  $h_{wu} h$

T  $\rightarrow$  H:  $\{k\}_{k_{wu}}$

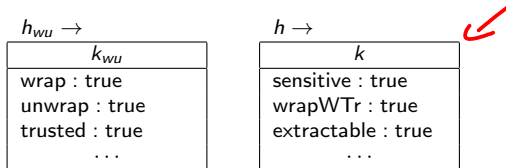
H  $\rightarrow$  T: UnwrapKey  $h_{wu} \{k\}_{k_{wu}}$  (sensitive, false)...

T  $\rightarrow$  H:  $h'$ , a handle to the new object:



Now read out key value!

## Attack 4: Downgrade *wrapWithTrusted*

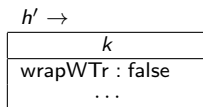


H  $\rightarrow$  T: WrapKey  $h_{wu} h$

T  $\rightarrow$  H:  $\{k\}_{k_{wu}}$

H  $\rightarrow$  T: UnwrapKey  $h_{wu} \{k\}_{k_{wu}}$  (wrapWTr, false) ...

T  $\rightarrow$  H:  $h'$ , a handle to the new object:



Now attack like key not protected by wrapWTr!

## Security Goal 3

'Bad role' for *trusted* keys:

- ▶ can be used for unwrapping into a key object of type other than *wrap-with-trusted*.

### Absence of this role for *trusted*

Every key  $k$  that is available with type *trusted* has never been and will never be enabled for the 'bad role'.

## Security Goal 3

'Bad role' for *trusted* keys:

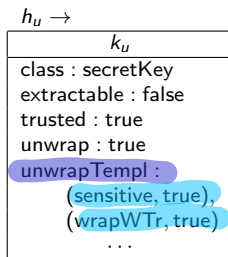
- ▶ can be used for unwrapping into a key object of type other than *wrap-with-trusted*.

### Absence of this role for *trusted*

Every key  $k$  that is available with type *trusted* has never been and will never be enabled for the 'bad role'.

How to achieve this?

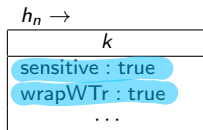
# Unwrap Templates



- ▶ If a key object has attribute `unwrapTempl` set to  $t$  then it can only unwrap into key objects that match  $t$ .

H  $\rightarrow$  T: UnwrapKey  $h_u \{k\}_{k_u}$  (sensitive, true), (wrapWTr, true), ..

T  $\rightarrow$  H:  $h_n$ , a handle to the new object:



# 1. How to set up trusted keys? Like this:

trusted : true
extractable : false
copyable : false
modifiable : false
decrypt : false
unwrapTempl :
sensitive : true
wrapWTr : true
...

For all Cryptoki tokens: when the SO installs trusted keys like this then:

1. Secrecy of *trusted*
2. Absence of role 'data decryption' for *trusted*
3. Absence of role 'unwrap into a key object with type other than wrap-with-trusted' for *trusted*

## 2. Additional constraints on functionality necessary? No!

trusted : true extractable : false copyable : false modifiable : false
decrypt : false
unwrapTempl : sensitive : true wrapWTr : true
...

For all Cryptoki tokens: when the SO installs trusted keys like this then:

- ▶ Origin-Secrecy of *wrap-with-trusted*.

# Other Secure Configurations

## Other Secure Configurations

Origin-Secrecy for sub-types of *wrap-with-trusted*?  
(E.g. enabled for data encryption or wrapping)

- ▶ yes, this is included.

## Other Secure Configurations

Origin-Secrecy for sub-types of *wrap-with-trusted*?  
(E.g. enabled for data encryption or wrapping)

- ▶ yes, this is included.

Secrecy of *wrap-with-trusted*?

When *trusted* keys are always symmetric:

- ▶ yes, but only with slight constraints on functionality.
- ▶ Reason: lineage is impossible to achieve without functionality constraints for all key types apart from *trusted*.

## Other Secure Configurations

Origin-Secrecy for sub-types of *wrap-with-trusted*?  
(E.g. enabled for data encryption or wrapping)

- ▶ yes, this is included.

Secrecy of *wrap-with-trusted*?

When *trusted* keys are always symmetric:

- ▶ yes, but only with slight constraints on functionality.
- ▶ Reason: lineage is impossible to achieve without functionality constraints for all key types apart from *trusted*.

When *trusted* keys are also public keys:

- ▶ No!
- ▶ Reason: lack of authenticated public key encryption in Cryptoki.

## Other Secure Configurations

Origin-Secrecy for sub-types of *wrap-with-trusted*?  
(E.g. enabled for data encryption or wrapping)

- ▶ yes, this is included.

Secrecy of *wrap-with-trusted*?

When *trusted* keys are always symmetric:

- ▶ yes, but only with slight constraints on functionality.
- ▶ Reason: lineage is impossible to achieve without functionality constraints for all key types apart from *trusted*.

When *trusted* keys are also public keys:

- ▶ No!
- ▶ Reason: lack of authenticated public key encryption in Cryptoki.

Key Separation similar.

# Approach

'Security-APIs are like protocols  
and they are also like programs'

*a New*

- complete*
1. Model the data structures of Cryptoki as abstract data types  
CASL (Common Algebraic Specification Language)
  2. Specify the commands and their policies of Cryptoki as conditional rewrite system over the data types
  3. Formulate axioms in FOLTL with past operators
  4. Tableau proof method for backwards analysis
  - 5. Simulation equivalence to transfer results

(Fröschle, Sommer FAST'10), (Fröschle, Sommer 11, full draft)

# Formal Model versus Real World

What does this say about my real token?

## Model versus PKCS#11 Standard

- ▶ Model is direct translation of standard.
- ▶ Only constraint: 'message data type' introduces Dolev-Yao constraints: abstracts away cryptographic attacks.
- ▶ Note: 'message data type' is plugin and can be refined.

## PKCS#11 Standard versus Real Token

- ▶ Compliance tests!

# Impact for Configurations

Configurations without functionality constraints:

- ▶ if token respects standard then token is secure modulo cryptographic attacks.
- ▶ axioms used in proofs highlight what is critical to check in compliance checks.
- ▶ e.g. attribute extractable becomes read-only once set to false.

Configurations with functionality constraints:

- ▶ Configuration will define clear compliance tests.
- ▶ e.g. can't use `CreateObject` to create *wrap-with-trusted* key.

Standardize Security Profiles for various Use Cases!

# Conclusions

- ▶ 'Wrap with trusted' and 'trusted' very good features.
- ▶ Wrap and unwrap templates very expressive but perhaps a bit difficult to understand and configure.
- ▶ Clear separation between logical structure and supported cryptographic mechanisms  
⇒ formal specification in about 4 pages.
- ▶ Mind Cryptoki's KeyDerive function!  
Serious attacks in (Sommer, Master thesis, 2009)

# Formal Specification