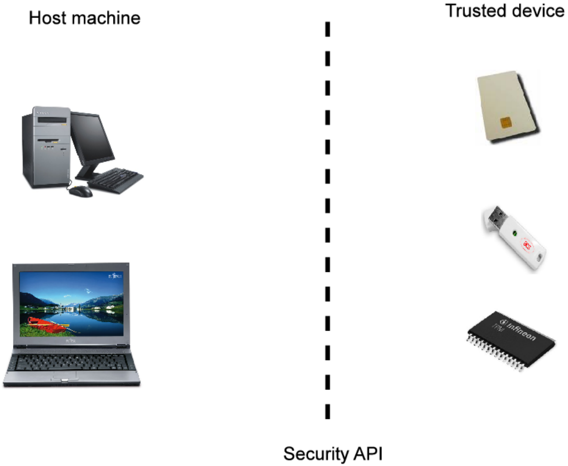
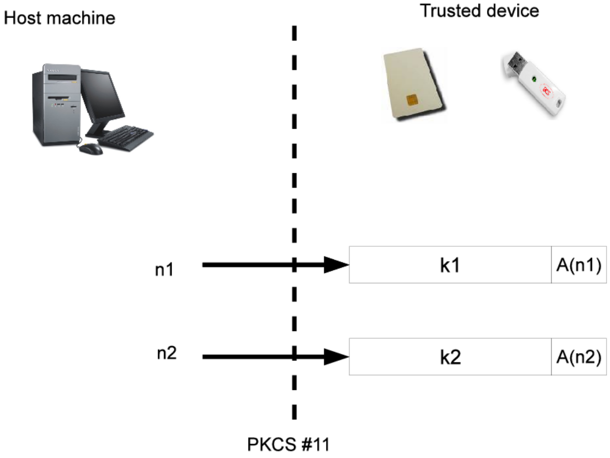
Attacking and fixing PKCS#11

Security Course, Ca' Foscari, 2016

Security APIs



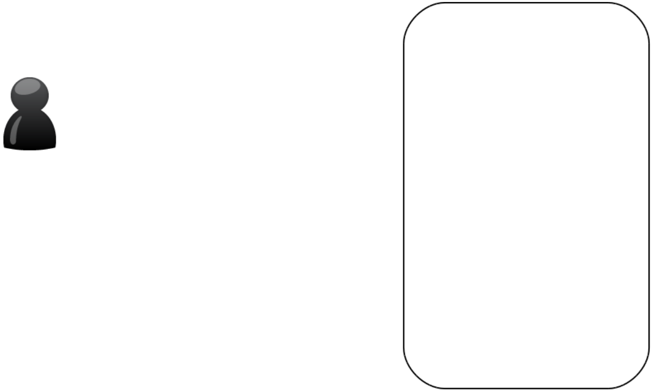
PKCS#11 API for trusted devices



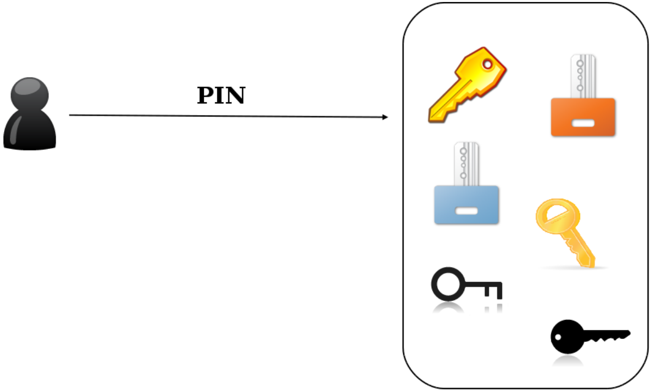
Outline

- API Overview
- PKCS#11 key management attacks
- API configuration problems
- How to make PKCS#11 secure?

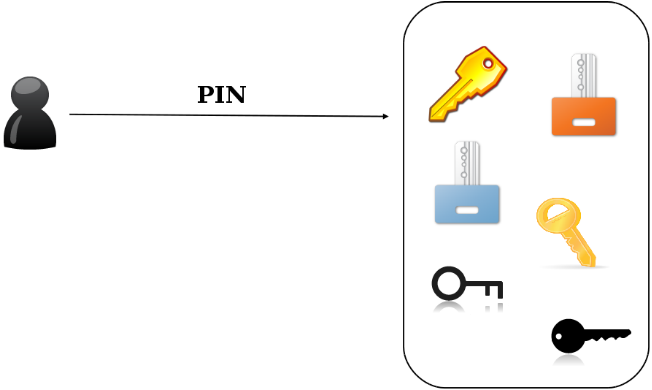
PKCS#11, an overview



PKCS#11, an overview

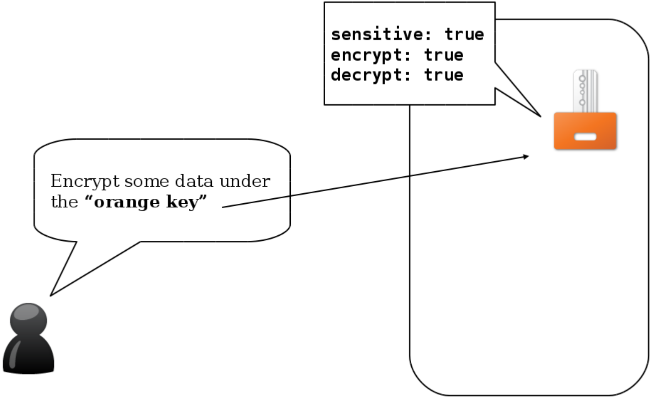


PKCS#11, an overview



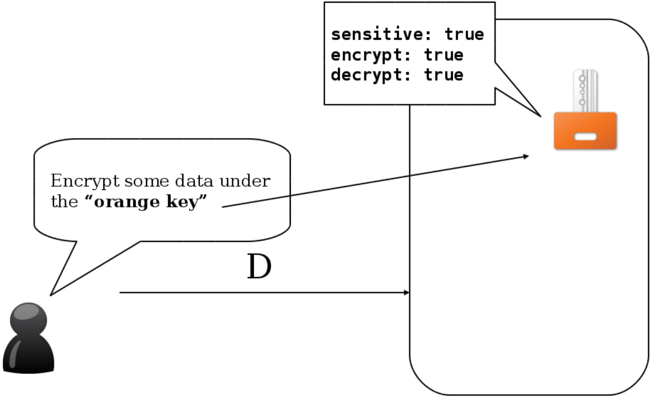
- the PIN is a ‘second-layer’ protection to unlock the token
  ⇒ it should never give access to `sensitive` key values

PKCS#11 keys and cryptographic operations



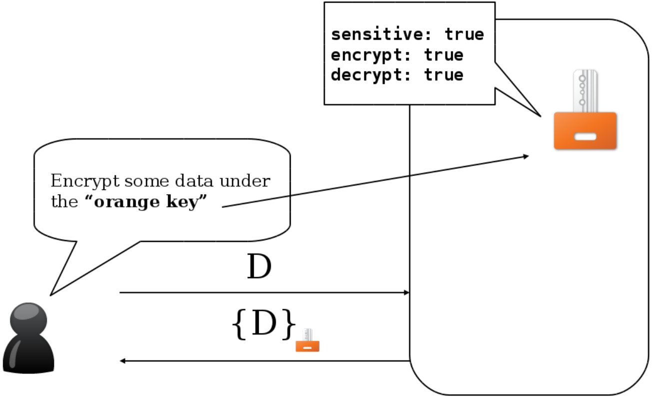
- Keys have *attributes* and are referenced via *handles*
- APIs for *cryptographic operations*

PKCS#11 keys and cryptographic operations



- Keys have *attributes* and are referenced via *handles*
- APIs for *cryptographic operations*

PKCS#11 keys and cryptographic operations



- Keys have *attributes* and are referenced via *handles*
- APIs for *cryptographic operations*

Security of keys

Confidentiality of sensitive keys

- `sensitive` keys never accessible as plaintext outside the device
  ... even if we know the PIN

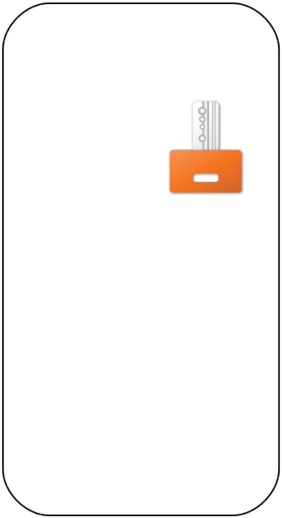
Attack scenario

1. token used on compromised host
2. attacker sniffs PIN and extracts `sensitive` keys
3. attacker clones the token

"... the PIN may be passed through the operating system. This can make it easy for a rogue application on the operating system to obtain the PIN ... " [RSA Security]

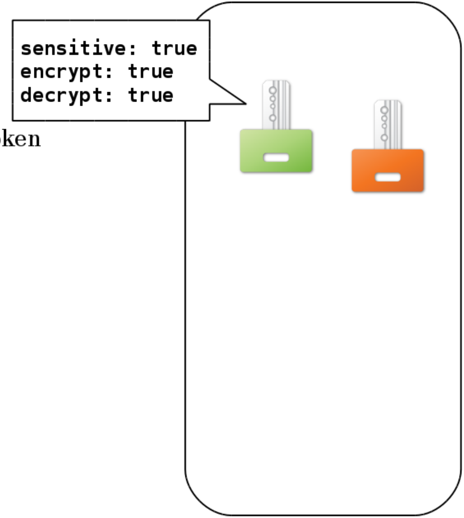
PKCS#11 key management

Create a new key inside the token

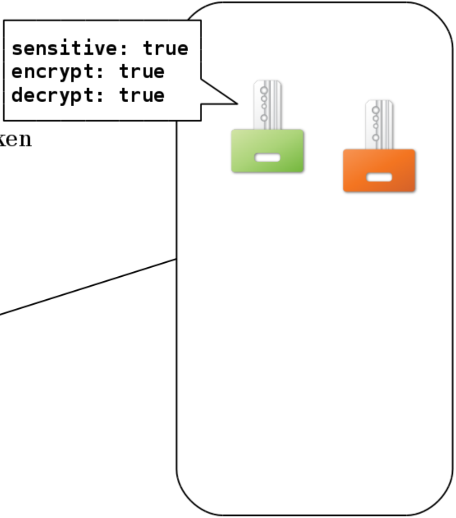


PKCS#11 key management

Create a new key inside the token

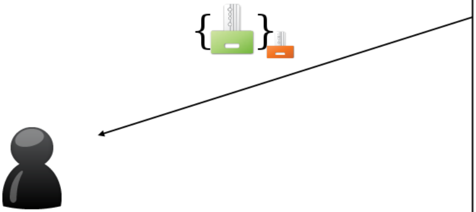


PKCS#11 key management



Create a new key inside the token

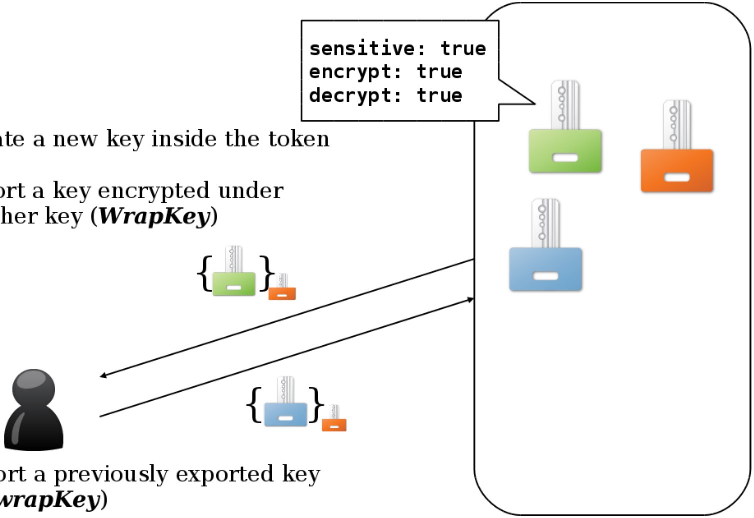
Export a key encrypted under another key (***WrapKey***)



PKCS#11 key management

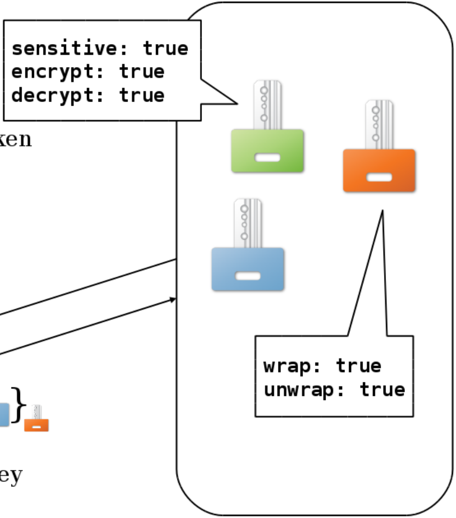
Create a new key inside the token

Export a key encrypted under another key (***WrapKey***)



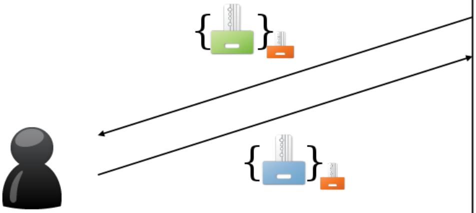
Import a previously exported key (***UnwrapKey***)

PKCS#11 key management



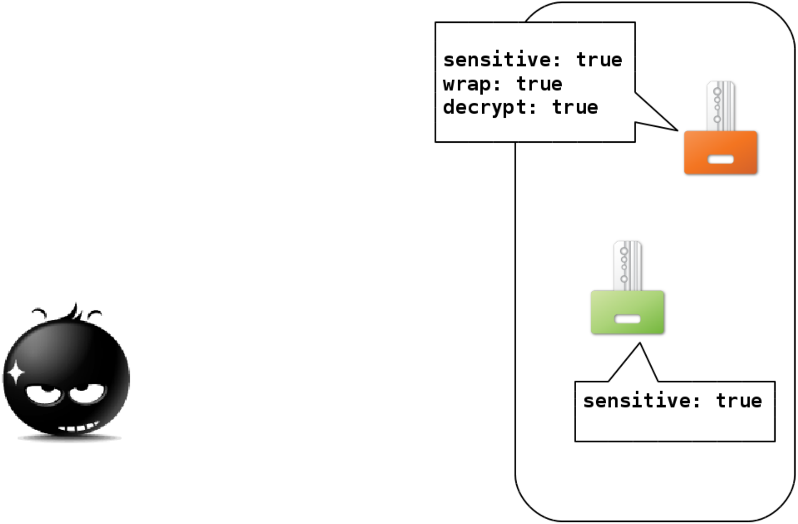
Create a new key inside the token

Export a key encrypted under another key (***WrapKey***)

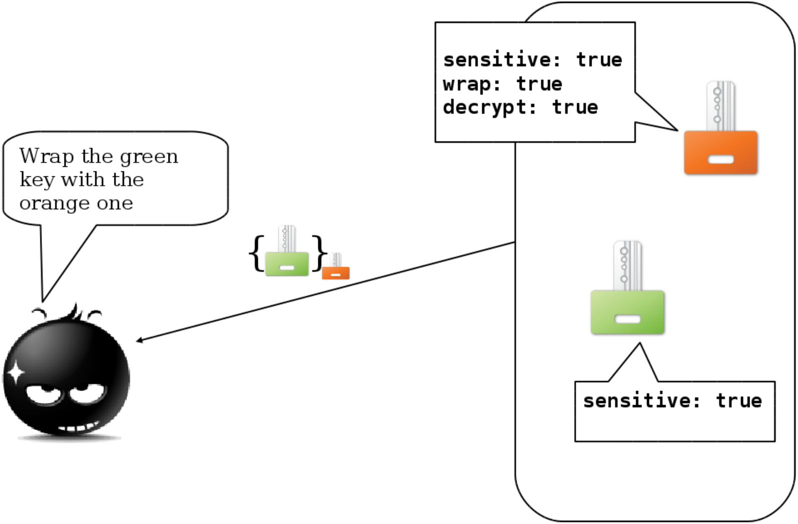


Import a previously exported key (***UnwrapKey***)

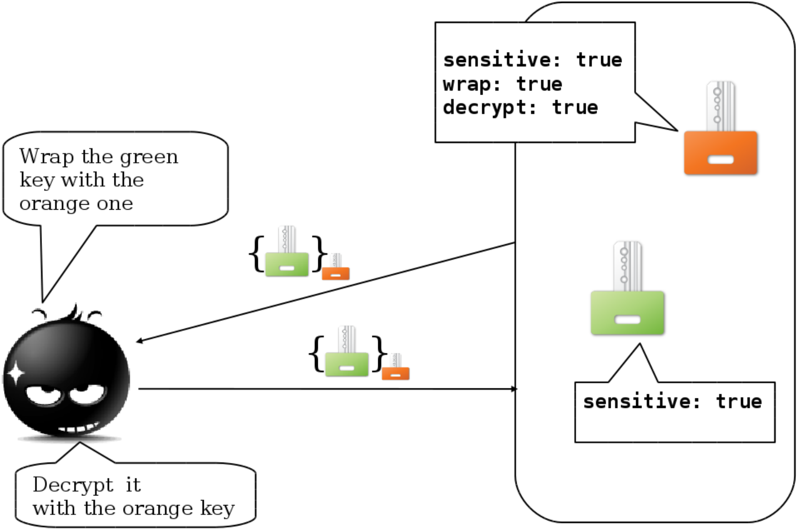
A simple API-level attack [Clulow CHES'03]



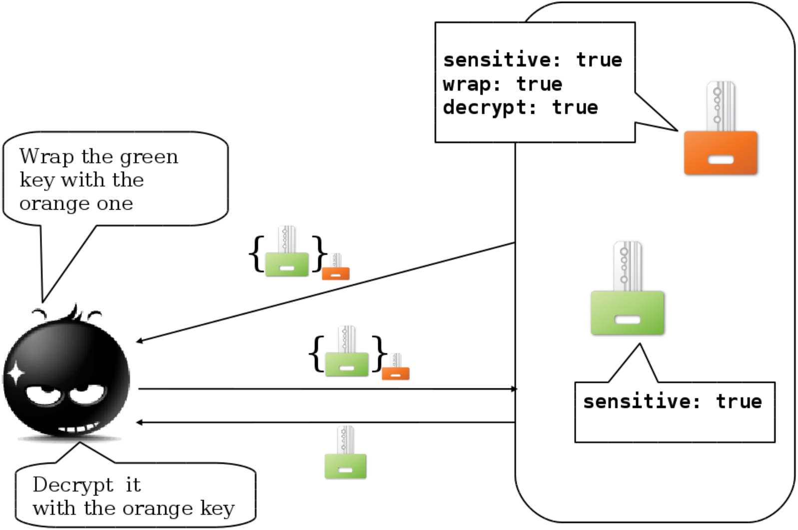
A simple API-level attack [Clulow CHES'03]



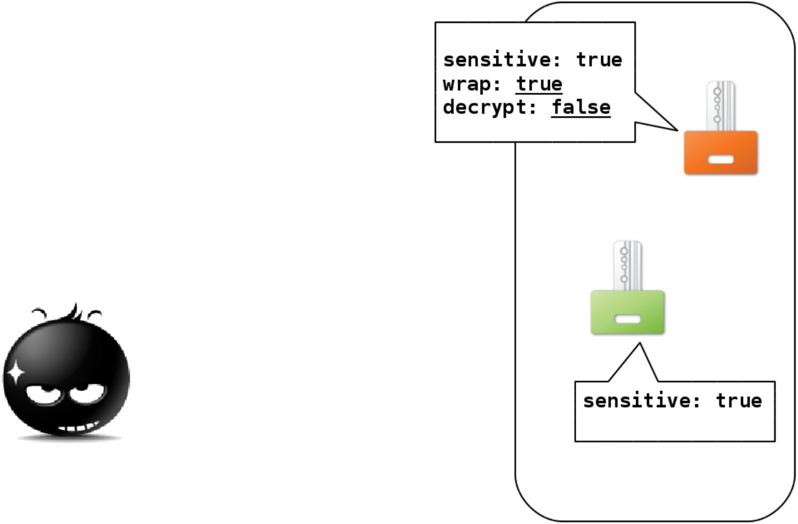
A simple API-level attack [Clulow CHES'03]



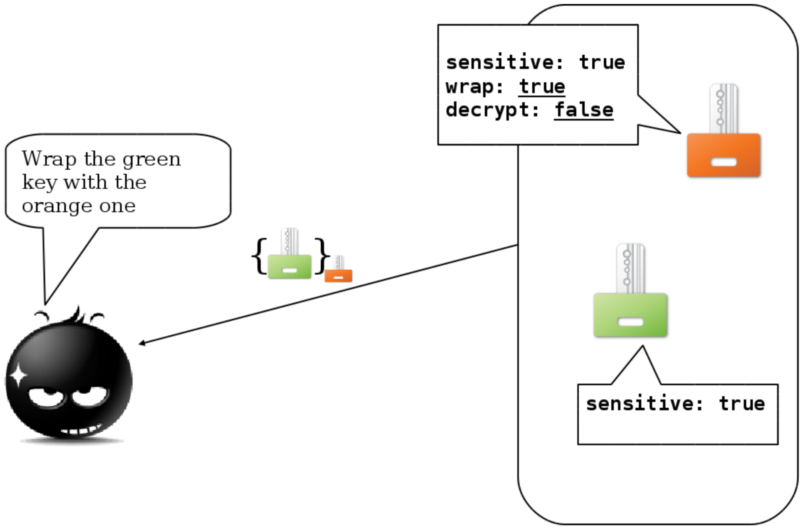
A simple API-level attack [Clulow CHES'03]



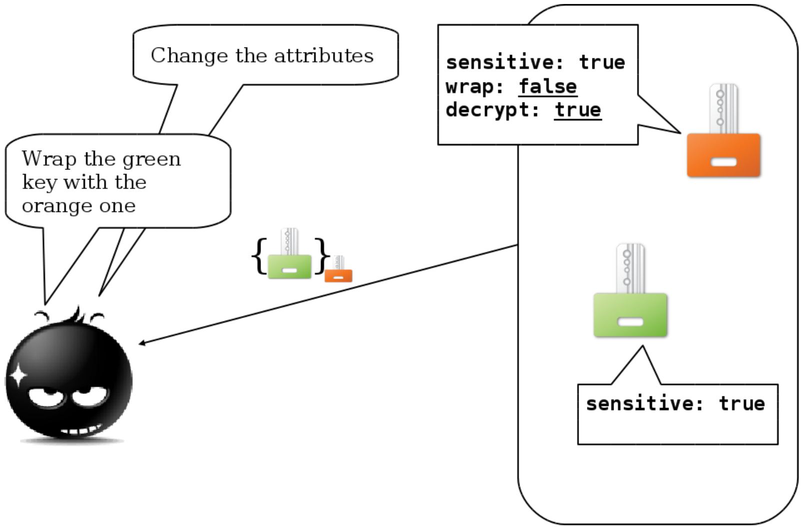
Key separation: forbid wrap and decrypt together



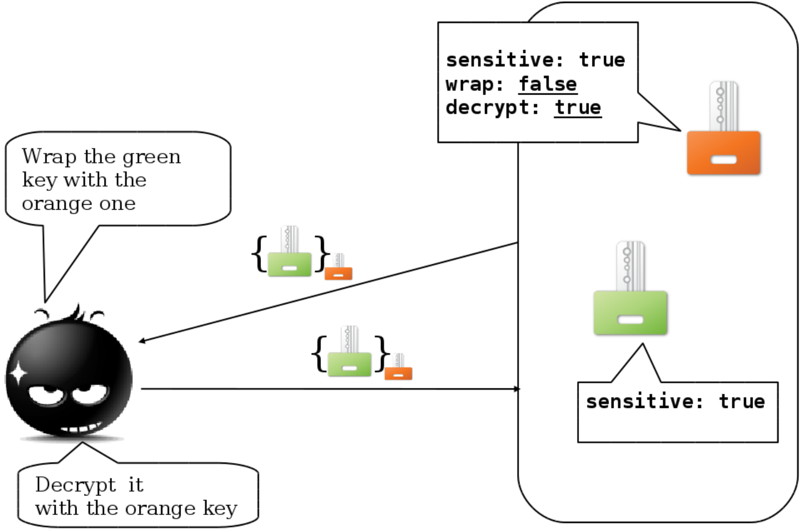
Key separation: forbid wrap and decrypt together



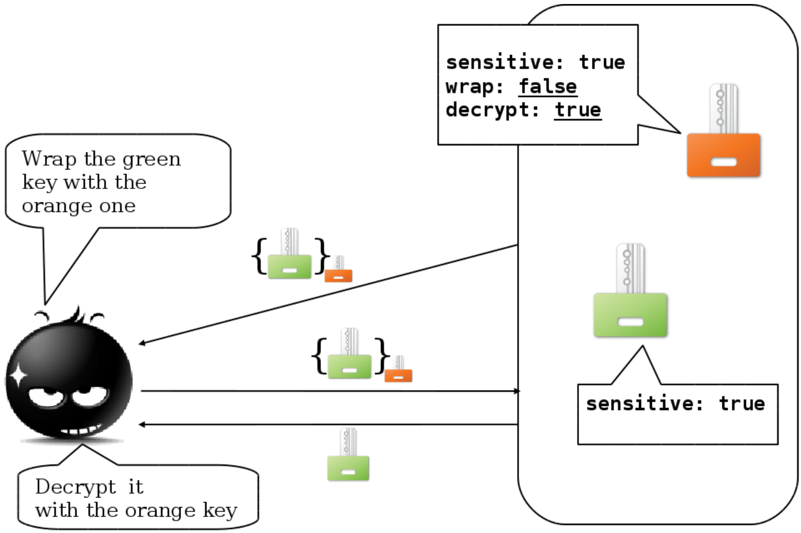
Key separation: forbid wrap and decrypt together



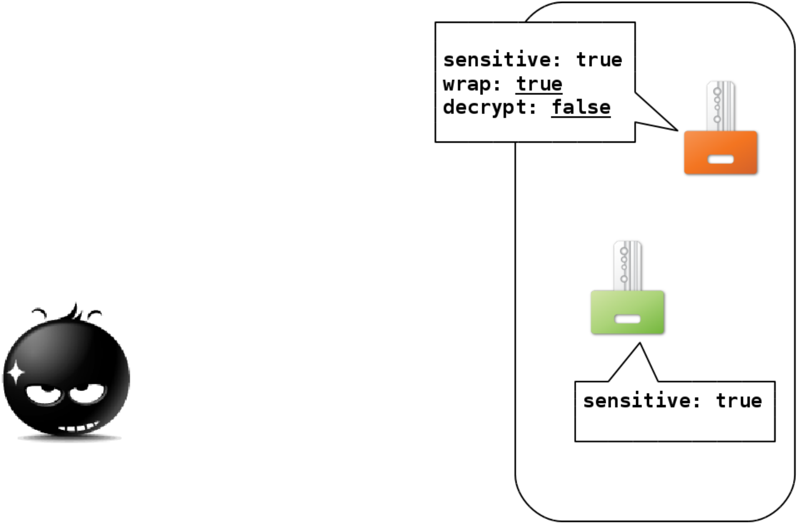
Key separation: forbid wrap and decrypt together



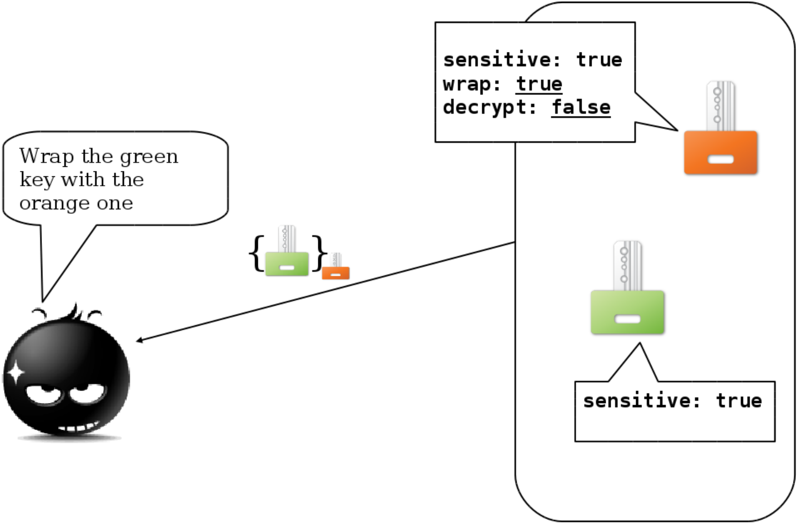
Key separation: forbid wrap and decrypt together



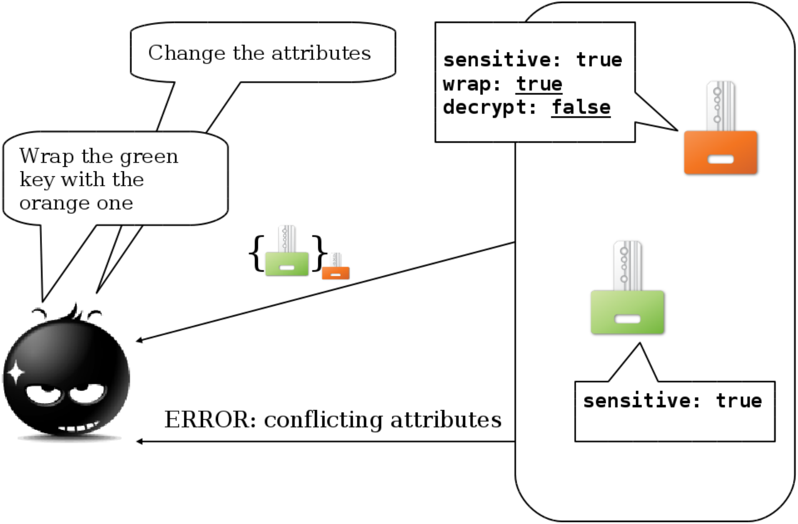
Well ... make attributes 'sticky on'



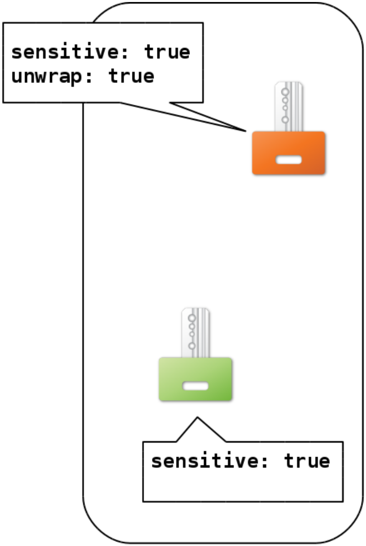
Well ... make attributes 'sticky on'



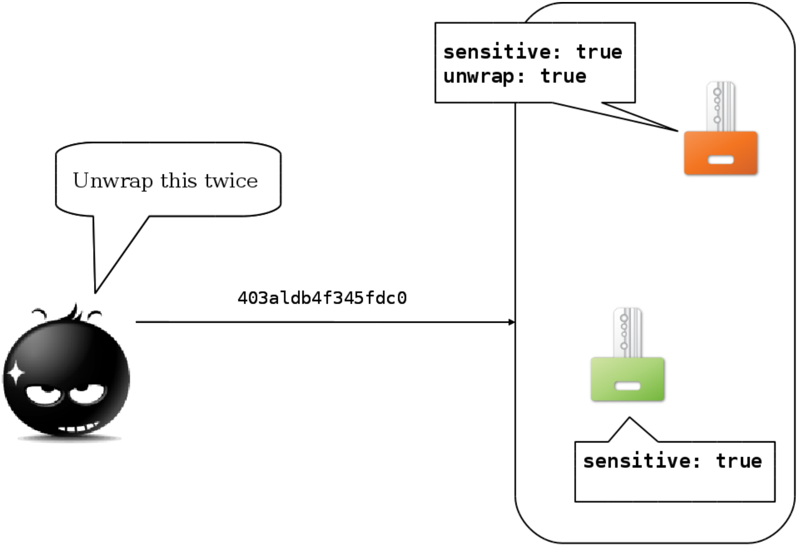
Well ... make attributes 'sticky on'



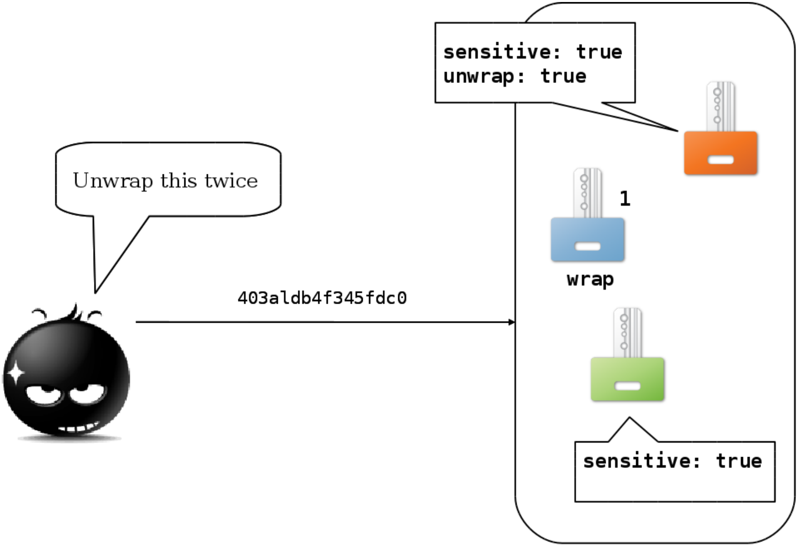
But still ...



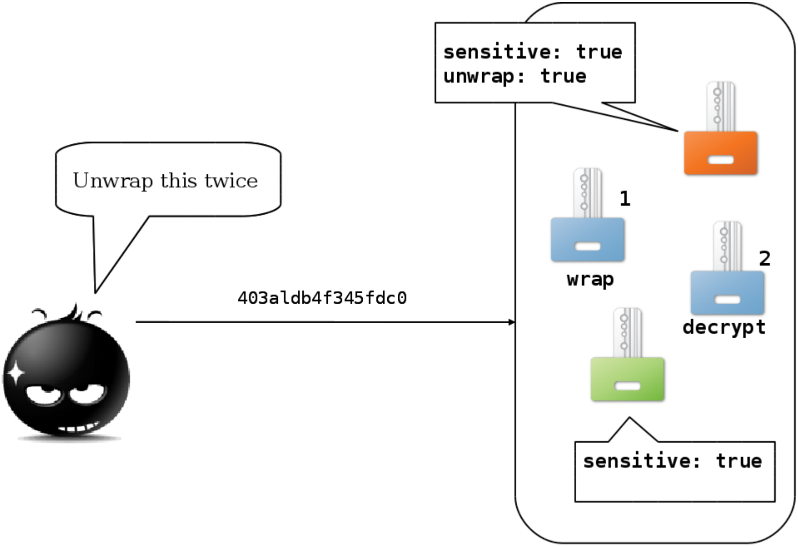
But still ...


sensitive: true
unwrap: true
Unwrap this twice
403a1db4f345fdc0
sensitive: true

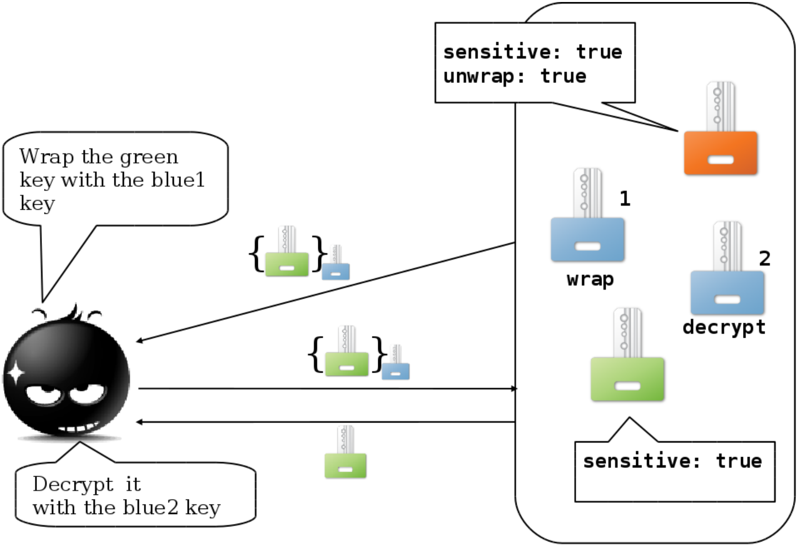
But still ...


Unwrap this twice
403a1db4f345fdc0
sensitive: true
unwrap: true
1
wrap
sensitive: true

But still ...


Unwrap this twice
403a1db4f345fdc0
sensitive: true
unwrap: true
1
wrap
2
decrypt
sensitive: true

But still ...


sensitive: true
unwrap: true
Wrap the green
key with the blue1
key
1
wrap
2
decrypt
sensitive: true
Decrypt it
with the blue2 key

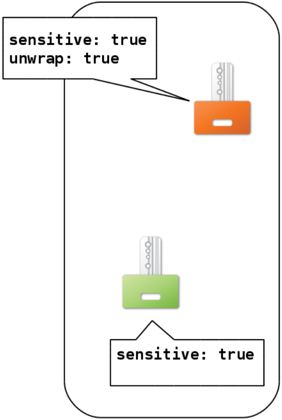
Now what?

Now what?

- check if two instances of the same key have different attributes
  - is this of any help?

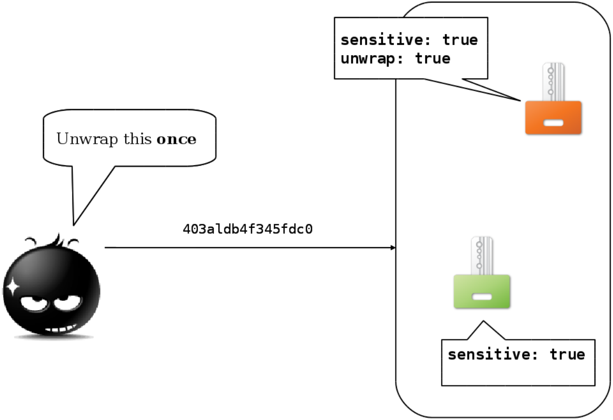
Now what?

- check if two instances of the same key have different attributes
  - is this of any help?



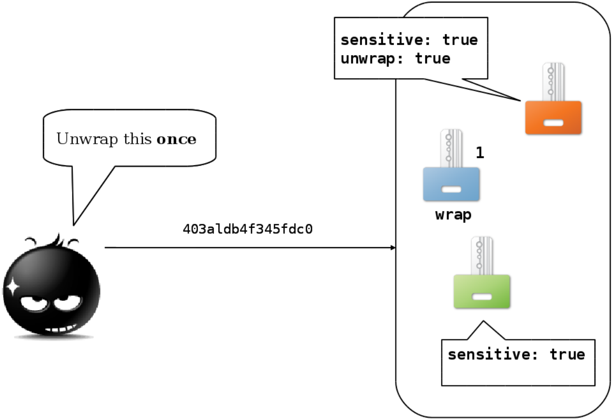
Now what?

- check if two instances of the same key have different attributes
  - is this of any help?



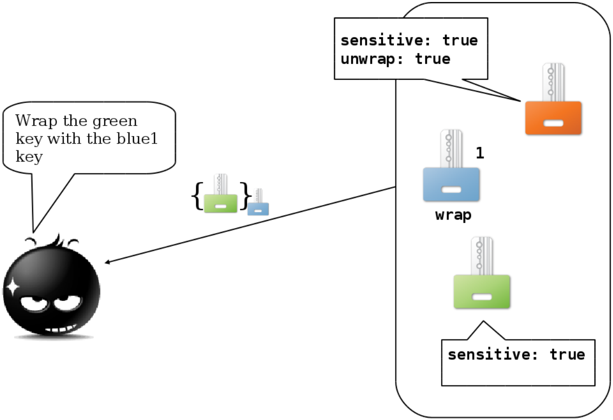
Now what?

- check if two instances of the same key have different attributes
  - is this of any help?



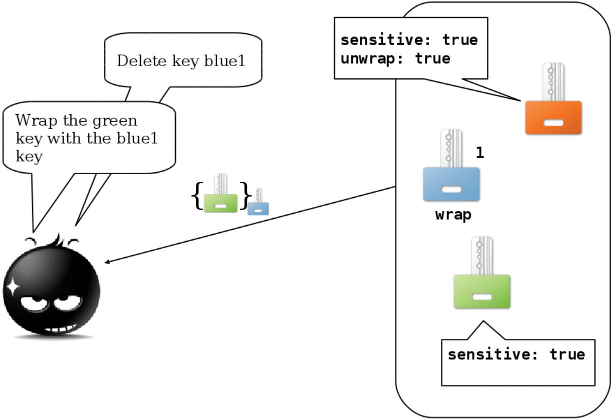
Now what?

- check if two instances of the same key have different attributes
  - is this of any help?



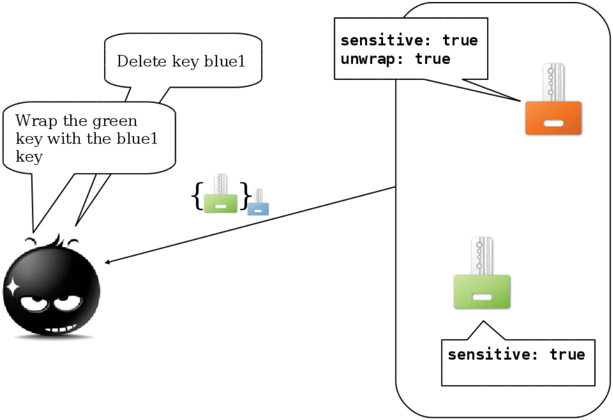
Now what?

- check if two instances of the same key have different attributes
  - is this of any help?



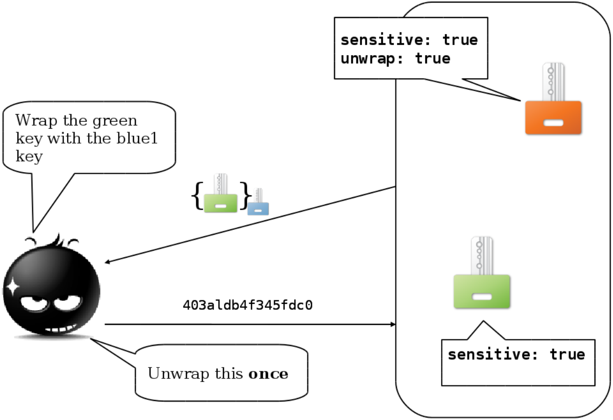
Now what?

- check if two instances of the same key have different attributes
  - is this of any help?



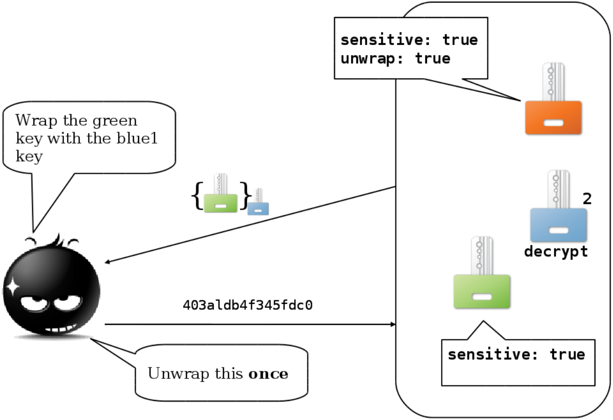
Now what?

- check if two instances of the same key have different attributes
  - is this of any help?



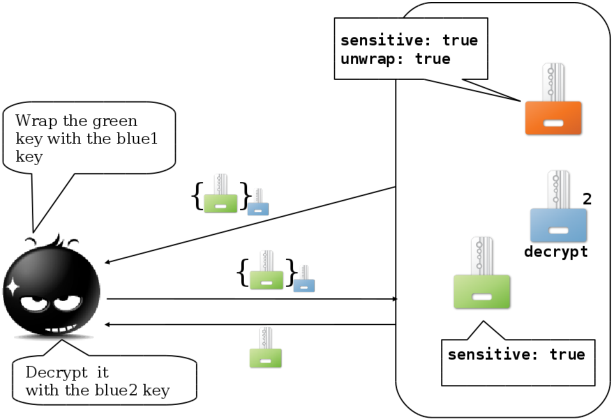
Now what?

- check if two instances of the same key have different attributes
  - is this of any help?



Now what?

- check if two instances of the same key have different attributes
  - is this of any help?



Wrapping format

- keep track of key template when wrapping it
- check that it corresponds when unwrapping

Wrapping format

- keep track of key template when wrapping it
- check that it corresponds when unwrapping

💡 Compute a CBC-MAC of the wrapped key together with its relevant attributes

$$\mathrm{MAC}_{k_m}(\{k_1\}_{k_2}, \text{sensitive}, \text{wrap}, \text{unwrap}, \ldots)$$

and give it as output together with $\{k_1\}_{k_2}$

- if the MAC does not correspond the key is not imported

Wrapping format

- keep track of key template when wrapping it
- check that it corresponds when unwrapping

💡 Compute a CBC-MAC of the wrapped key together with its relevant attributes

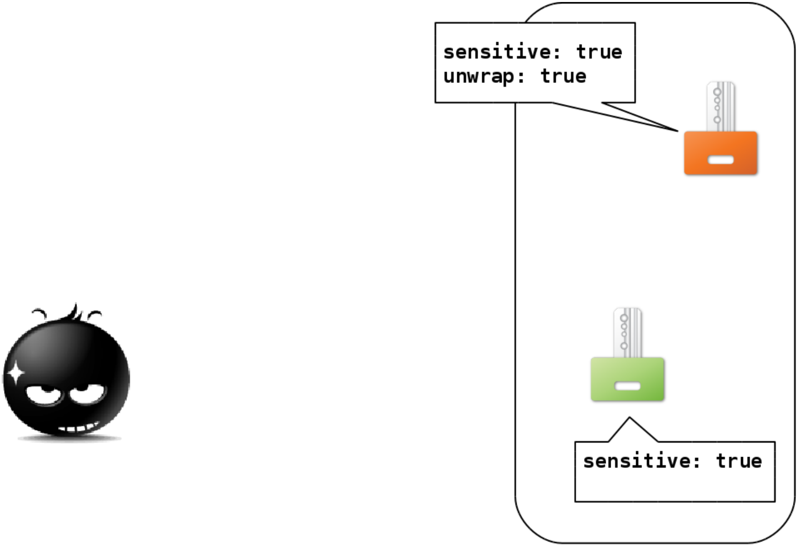
$$\mathsf{MAC}_{k_m}(\{k_1\}_{k_2}, \mathsf{sensitive}, \mathsf{wrap}, \mathsf{unwrap}, \ldots)$$

and give it as output together with $\{k_1\}_{k_2}$

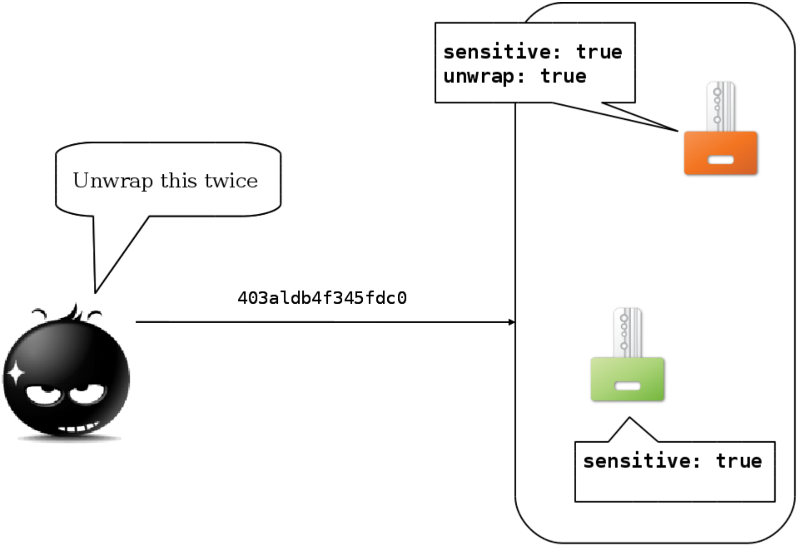
- if the MAC does not correspond the key is not imported

Note: k_m can be derived from k_2, e.g., by encrypting some constant

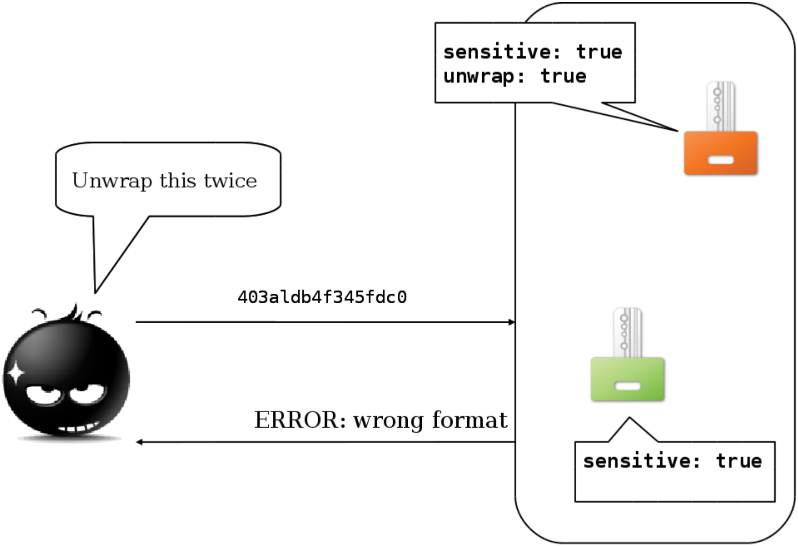
Unwrap of arbitrary data is prevented



Unwrap of arbitrary data is prevented



Unwrap of arbitrary data is prevented



Summary: Attribute policies and wrapping formats

Sticky

Once an attribute is set (unset), it may not be unset (set).
Read-only attributes can be thought as both sticky on and off.

Conflicting

Pairs of attributes that cannot be simultaneously set.
(not in the PKCS#11 documentation)

Wrapping format

Keep track of relevant attributes when wrapping, and check they are the same when unwrapping

What's the problem?

What’s the problem?

buffalo buffalo buffalo buffalo buffalo buffalo buffalo buffalo

What’s the problem?

buffalo buffalo buffalo buffalo buffalo buffalo buffalo buffalo

Buffalo buffalo Buffalo buffalo buffalo buffalo Buffalo buffalo

What's the problem?

buffalo buffalo buffalo buffalo buffalo buffalo buffalo buffalo

Buffalo buffalo Buffalo buffalo buffalo buffalo Buffalo buffalo

Buffalo buffalo, Buffalo buffalo buffalo, buffalo Buffalo buffalo

What’s the problem?

buffalo buffalo buffalo buffalo buffalo buffalo buffalo buffalo

Buffalo buffalo Buffalo buffalo buffalo buffalo Buffalo buffalo

Buffalo buffalo, Buffalo buffalo buffalo, buffalo Buffalo buffalo

buffalo FROM Buffalo WHO buffalo (intimidate) buffalo FROM Buffalo, buffalo (intimidate) buffalo FROM Buffalo

Formal analysis of PKCS#11 (2 slides!)

- Terms representing keys, ciphertexts, handles

$$k, senc\,(d, k)\,, h(n, k)$$

- Rules $T; L \xrightarrow{\text{new } \tilde{n}} T'; L'$ representing API calls

$$h\,(x_1, y_1)\,, y_2;\ \texttt{encrypt}\,(x_1)\ \rightarrow\ senc\,(y_2, y_1)$$

- Transitions $(S, V) \rightsquigarrow (S', V')$ representing API invocation

$$\langle\ \{h(n, k), d\}; \texttt{encrypt}(n)\ \rangle \rightsquigarrow \langle\ \{h(n, k), d, senc(d, k)\}; \texttt{encrypt}(n)\ \rangle$$

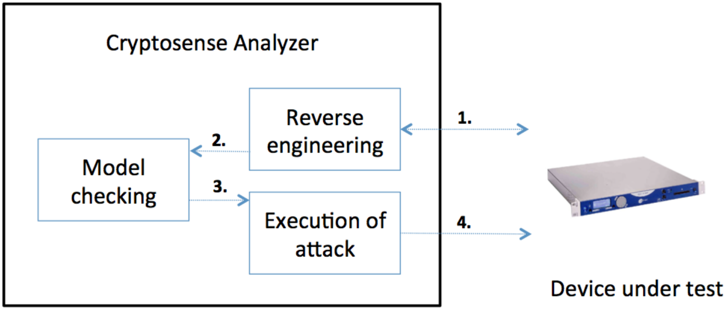
Wrap-Decrypt attack, formally

- Rules for key generation, wrap, decrypt:

$$\begin{array}{rcl} & \xrightarrow{\text{new } n,k} & h(n,k)\,;\mathcal{A} \\ h(x_1,y_1)\,,h(x_2,y_2)\,;\mathtt{wrap}(x_1)\,,\mathtt{extract}(x_2) & \longrightarrow & senc(y_2,y_1) \\ h(x_1,y_1)\,,senc(y_2,y_1)\,;\mathtt{decrypt}(x_1) & \longrightarrow & y_2 \end{array}$$

- We start from state $\langle\ \{h(n_1,k_1)\}, \mathtt{sensitive}(n_1), \mathtt{extract}(n_1)\ \rangle$
  - $\leadsto \langle\ \{h(n_1,k_1), h(n_2,k_2)\},$ $\mathtt{sensitive}(n_1), \mathtt{extract}(n_1), \mathtt{wrap}(n_2), \mathtt{decrypt}(n_2)\ \rangle$
  - $\leadsto \langle\ \{h(n_1,k_1), h(n_2,k_2), senc(k_1,k_2)\},$ $\mathtt{sensitive}(n_1), \mathtt{extract}(n_1), \mathtt{wrap}(n_2), \mathtt{decrypt}(n_2)\ \rangle$
  - $\leadsto \langle\ \{h(n_1,k_1), h(n_2,k_2), senc(k_1,k_2), k_1\},$ $\mathtt{sensitive}(n_1), \mathtt{extract}(n_1), \mathtt{wrap}(n_2), \mathtt{decrypt}(n_2)\ \rangle$

Crytpsense Analyzer



Results of testing of Tokens/Smarcards

Device		Supported Functionality						Attacks found				
Brand	Model	s	as	cobj	chan	w	ws	wd	rs	ru	su	Tk
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

Towards a secure PKCS#11

Fixing this kind of attacks is far from being trivial

- most real tokens are either vulnerable or cut-down so to avoid wrap/unwrap [Bortolozzo et al. CCS'10].
- secure configuration + wrapping format [Delaune et al. JCS'09] (e.g., Eracom mechanism `Wrapkey_DES3_CBC`, out of the standard)
- more on wrapping formats [Fröschle, Steel ARSPA-WITS'09]
- configuration with mixed roles [Bortolozzo et al. CCS'10]
- configuration with trusted keys [Fröschle, Sommer FAST'11]
- key diversification [Centenaro et al. POST'12]

Summary

- Crypto APIs are irritatingly liberal
- Attacks to compromise a sensitive key and fixes
- The APIs are hard, sometimes impossible, to configure securely
- Formal methods are effective
  - Find (new) attacks
  - Provide a high level of certification

References

- M. Bortolozzo, M. Centenaro, R. Focardi, G. Steel.
  Attacking and Fixing PKCS#11 Security Tokens. ACM CCS'10.
- M. Bortolozzo, M. Centenaro, R. Focardi, G. Steel.
  CryptokiX: a cryptographic software token with security fixes. ASA'10.
- M. Centenaro, R. Focardi, F. Luccio.
  Type-based Analysis of PKCS#11 Key Management. POST 2012.
- J. Clulow. On the security of PKCS#11. CHES'03.
- S. Delaune, S. Kremer, G. Steel.
  Formal analysis of PKCS#11 and proprietary extensions. JCS 2009.
- Falcone, A., Focardi R.
  Formal Analysis of Key Integrity in PKCS#11. ARSPA-WITS'10.
- S. Fröschle and G. Steel. Analysing PKCS#11 Key Management APIs with Unbounded Fresh Data. ARSPA-WITS'09.
- S. Fröschle and N. Sommer
  Concepts and Proofs for Configuring PKCS#11. FAST'11
- RSA Security Inc. PKCS #11 v.2.20: Cryptographic Token Interface Standard. June 2004