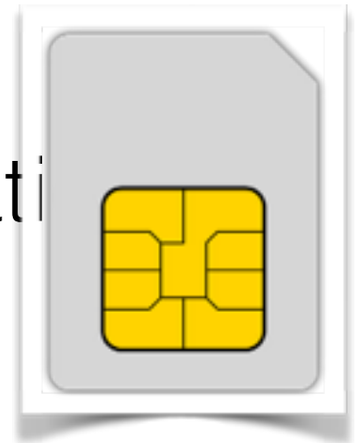
Automated backward analysis of PKCS#11 v2.20

Robert Künnemann — Technische Universität Darmstadt

EC SPRIDE
EUROPEAN CENTER FOR SECURITY AND PRIVACY BY DESIGN

PKCS#11

- defin raphic operati

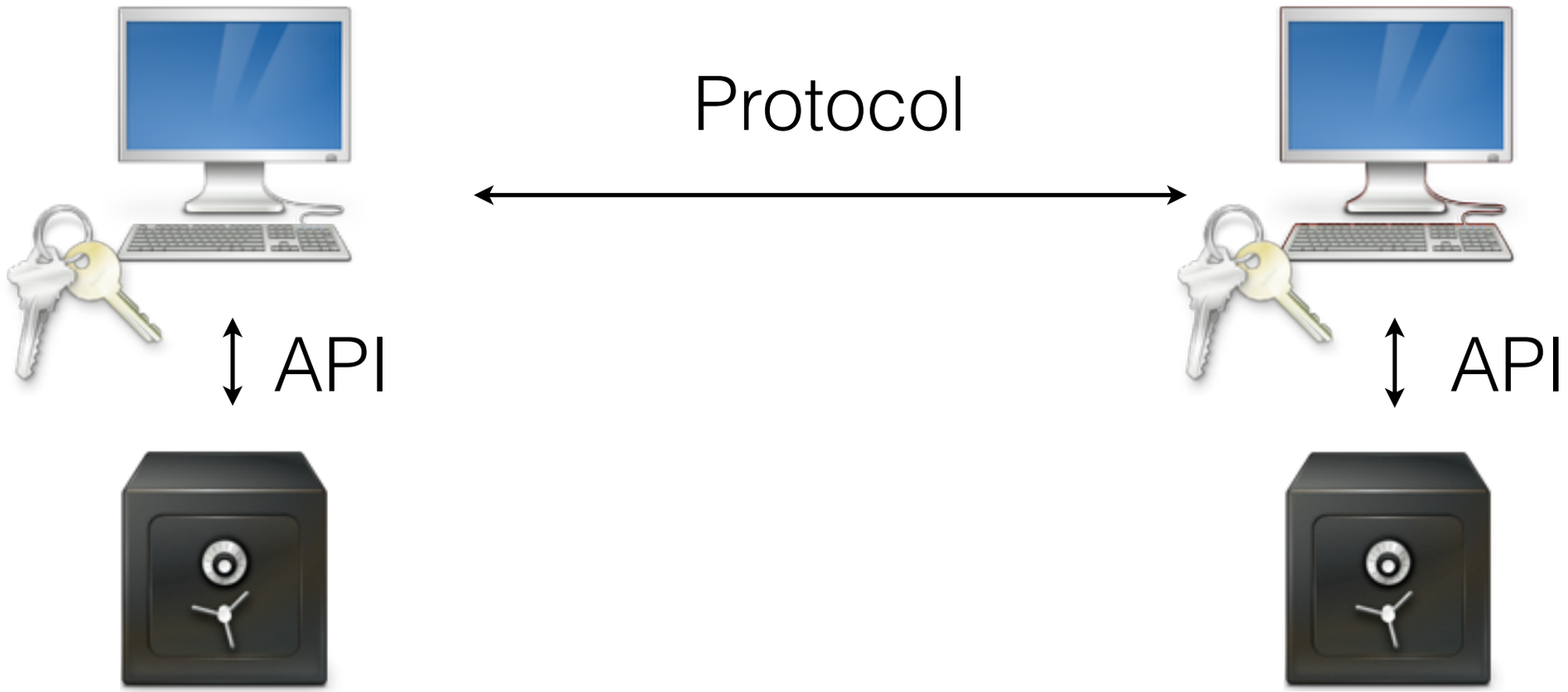


- used

- Hardware Security Modules (HSMs)
- Smart cards
- Software implementations, e.g. in Firefox

PKCS#11

- goal: protect cryptographic material:



PKCS#11

- goal: protect cryptographic material:
- corollary: must contain keys and implement cryptographic functions
- indirect access via handles
- security property: "sensitive" keys cannot be learned, even by corrupted parties

PKCS#11 (core)

- key-usage:
  - symmetric encryption/decryption
  - etc.
- key-management:
  - creation of keys
  - export (wrapping) and import (unwrapping)
- handles map to keys x templates (= set of attributes, e.g. enc, dec, wrap, unwrap, sensitive,..)

Contributions

- formal model of PKCS#11v2.20 in cryptographic process calculus with state (Dolev-Yao model)

- policy that allows for secure backup of usage-keys
  - secrecy of sensitive keys established using backward analysis
  - less automated, but more flexible than previous models

- proof finding heuristics for PKCS#11

Clulow's attack

set_wrap(h)

wrap(h,h)

c= $\{|k|\}_k$

set_decrypt(h)

decrypt(h,c)

k !!

Policies

- incomplete implementation of PKCS#11

- typically restricts creation/import of keys (templates) and attribute changes

PKCS#11 v2.20

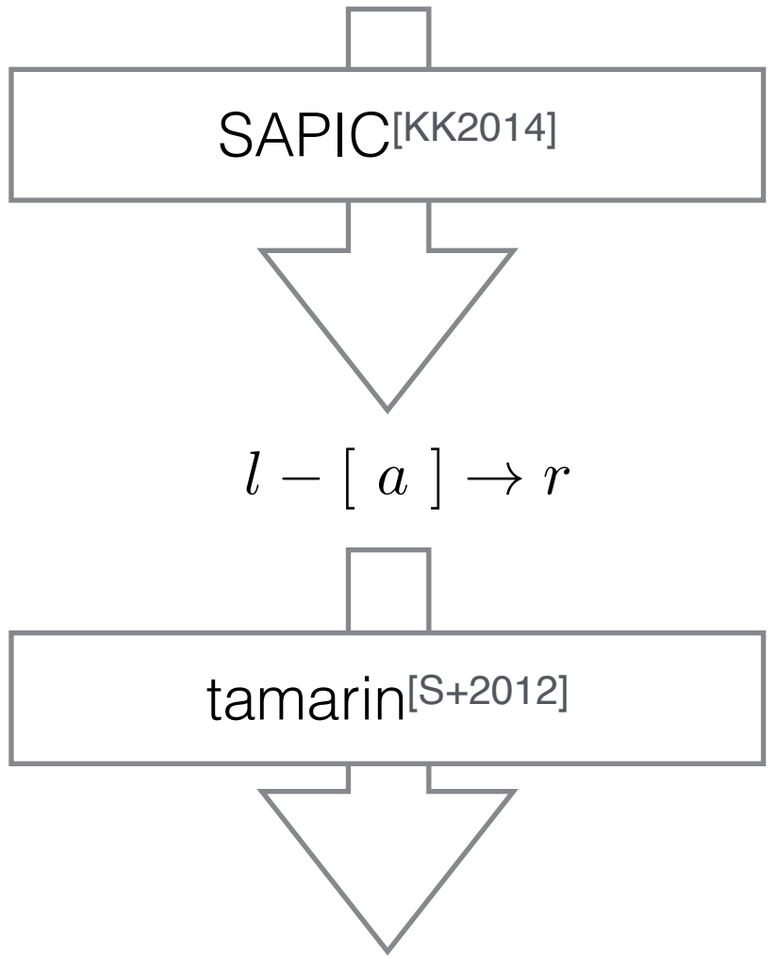
- new attributes:
  - **wrap-template:** wrap k under k_w
    template of k have to match k_w's wrap template
  - **unwrap-template:** import key in c using k_w
    handle to new key will have template matching k_w's unwrap template
  - recursive: templates contain attributes wrap-template and unwrap-template

Policy (simplified)

name	wrap/unwrap	enc/dec	sensitive	wt/ut
trusted	•		•	usage
usage		•	•	-
untrusted		•		-

Approach

$P_{init}; !(P_{create} \mid P_{dec} \mid P_{enc} \mid P_{wrap} \mid P_{unwrap} \mid P_{get\_keyval})$ + helping lemmas



- backward-analysis
- interactive
- automatic w/ custom heuristics

Modelling

$P_{init}; !(P_{create} \mid P_{dec} \mid P_{enc} \mid P_{wrap} \mid P_{unwrap} \mid P_{get\_keyval})$

$P_{wrap} := \mathsf{in}(\langle h1,h2 \rangle);$

Verification

- drawback: helping lemmas need to be written (but are verified automatically)

- standard 'smart' heuristic fails, e.g. wastes time on deduction of handles

- heuristics adapted to use case, e.g. resolve template lookups first

- optimisations for SAPIC output in general, e.g. resolve unlock operations and previous states right away

Proof

- drawback: helping lemmas need to be written (but are verified automatically)

1. message obtained by decryption was input by adversary OR a bad thing happened

2. imported (unwrapped) keys were once created on device OR a bad thing happened

3. if one bad thing happened, a worse thing happened before

Evaluation

	interaction	
lemma	no heuristics	heuristics
dec_limits	11	0
bad_keys	0	0
no key is wrap+dec	15	0
no key is enc+unwrap	29	0
cannot obtain key	6	0

Related work

Three major lines of work:

1. protocol verification approach
2. program verification approach
3. type-checking approach

- security token is (sole) participant in protocol
- early results using model checking [DKS2010],soundness for static policies [FS2009,B+2010]
- soundness result is model-specific (e.g. cannot deal with v2.20)

Related work

Three major lines of work:

1. protocol verification approach
2. program verification approach
3. type-checking approach

- first-order linear time logic with past operators [FS2010]
- manual (tableau) proofs, backward-analysis
- but: support for wrap/unwrap templates [FS2011]

Related work

Three major lines of work:

1. protocol verification approach
2. program verification approach
3. type-checking approach

- static analysis on C-implementation of token[CFL2013]
- generalised version that maps to PKCS#11 v2.20[AFL2013] with similar policy

Limitations & Future work

- raise degree of automation
  - synthesise lemmas (hard)
    - derive "general" lemmas (heuristics may help)
- policy can be expressed without v2.20 features, and was proven secure before [AFL2013] (using type-checking)
  - try approach on "real" dynamic policy
  - helping lemmas need to be altered

Conclusion

- backward-analysis approach can be automated (or protocol analysis approach can be made more precise)

- flexible and expressive modelling language in SAPIC, precise analysis with tamarin (msr)

- no model-specific soundness results needed

- possibility of analysing "real" dynamic policies for PKCS#11

Thank you for your attention.

References

[DKS2010] Stéphanie Delaune, Steve Kremer, and Graham Steel. “Formal Analysis of PKCS#11 and Proprietary Extensions”. In: Journal of Computer Security 18.6 (Nov. 2010).

[FS2009] Sibylle Fröschle and Graham Steel. “Analysing PKCS#11 Key Management APIs with Unbounded Fresh Data”. In: Joint Workshop on Automated Reasoning for Security Protocol Analysis and Issues in the Theory of Security (ARSPA-WITS’09). Vol. 5511. LNCS. Springer, 2009.

[B+2010] Matteo Bortolozzo et al. “Attacking and Fixing PKCS#11 Security Tokens”. In: CCS 2010. ACM Press, 2010.

[FS2010] Sibylle B. Fröschle and Nils Sommer. “Reasoning with Past to Prove PKCS#11 Keys Secure”. In: FAST 2010. Vol. 6561. LNCS. Springer, 2010.

[FS2011] Sibylle Fröschle and Nils Sommer. “Concepts and Proofs for Configuring PKCS#11”. In: FAST 2011. Vol. 7140. LNCS. Leuven, Belgium: Springer, 2012.

[CFL2013] Matteo Centenaro, Riccardo Focardi, and Flaminia L. Luccio. “Type-based analysis of key management in PKCS#11 cryptographic devices”. In: Journal of Computer Security 21.6 (2013).

[AFL2013] Pedro Adão, Riccardo Focardi, and Flaminia L. Luccio. “Type-Based Analysis of Generic Key Management APIs”. In: CSF 2013. IEEE, 2013, pp. 97–111.

[KK2014]Steve Kremer and Robert Künnemann. “Automated analysis of security protocols with **global** state”. In: Security and Privacy. IEEE Computer Society, 2014.

[S+2012]Benedikt Schmidt et al. “Automated Analysis of Diffie-Hellman Protocols and Advanced Security Properties”. In: CSF 2012. IEEE, 2012.

PKCS#11 (core)

- key-usage:
  - symmetric/~~asymmetric~~ encryption/decryption
  - ~~signatures, MAC, random number generation~~ …
- key-management:
  - creation ~~and (unencrypted) import~~ of keys
  - export (wrapping) and import (unwrapping)
  - ~~key derivation~~
- keys have attributes: enc,dec,wrap,unwrap,sensitive,..