ISACA 2015 K. WAYNE SNIPES AWARD HONORABLE MENTION

ISACA 2015 COMMUNICATIONS COMMENDATION

ISACA
Trust in, and value from, information systems
Venice Chapter

IV Conference on Application Security and Modern Technologies

In collaborazione con

OWASP
The Open Web Application Security Project

Università Ca'Foscari Venezia
Dipartimento di Scienze Ambientali Informatica e Statistica

Venezia, Università Ca' Foscari

23 Settembre 2016

APDU-Level Attacks in PKCS#11 Devices

Francesco Palmarini
Joint work with
Claudio Bozzato Riccardo Focardi Graham Steel

Università Ca' Foscari, Venice, Italy

September 23, 2016

Talk Outline

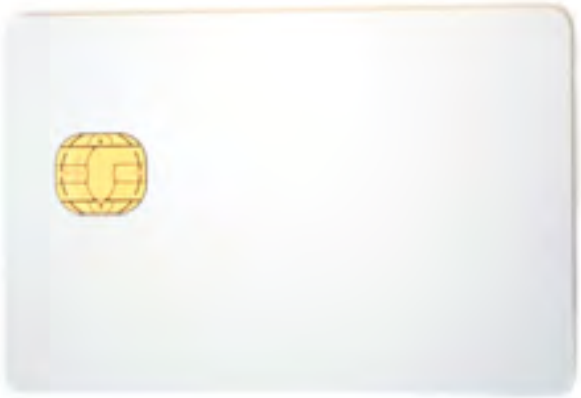
Agenda

1. Background
2. Architecture of PKCS#11 devices
3. Threat model
4. APDU-level attacks on real devices
5. Fixes and mitigations

The problem

Cryptographic hardware

Cyrptographic hardware allows cryptographic operations to be performed inside a protected, tamper-resistant environment



PKCS#11 security properties

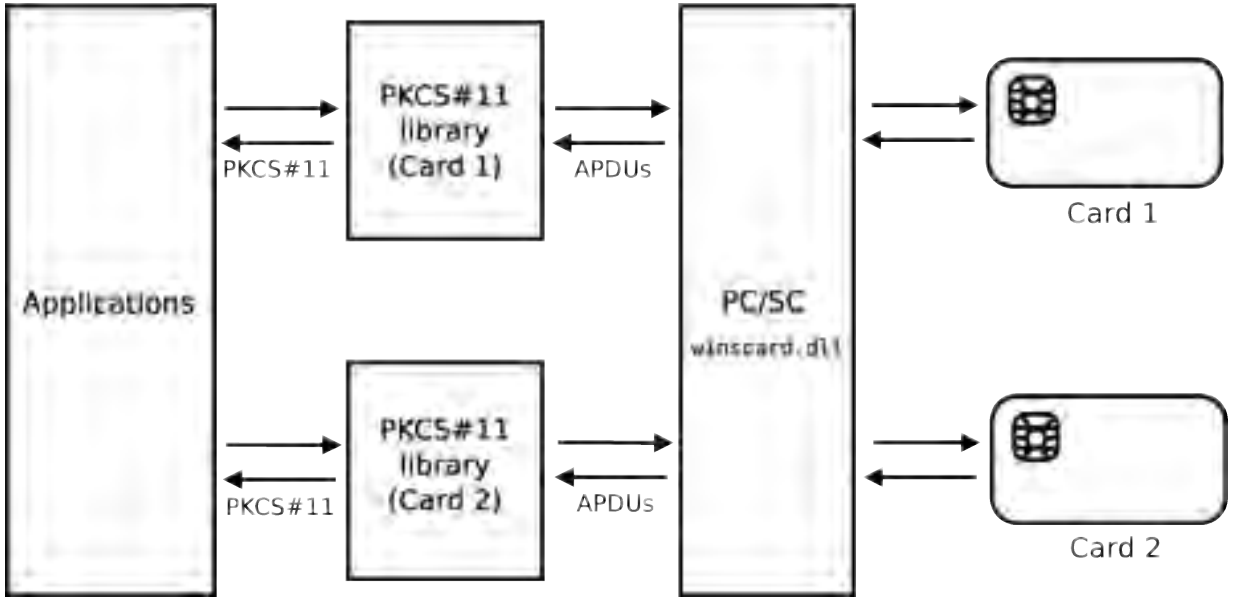
PKCS#11 is a standard API for cryptographic devices:

- Crypto operations should be performed inside the device
- Sensitive keys should never be leaked as plaintexts

Background

PKCS#11 middleware architecture

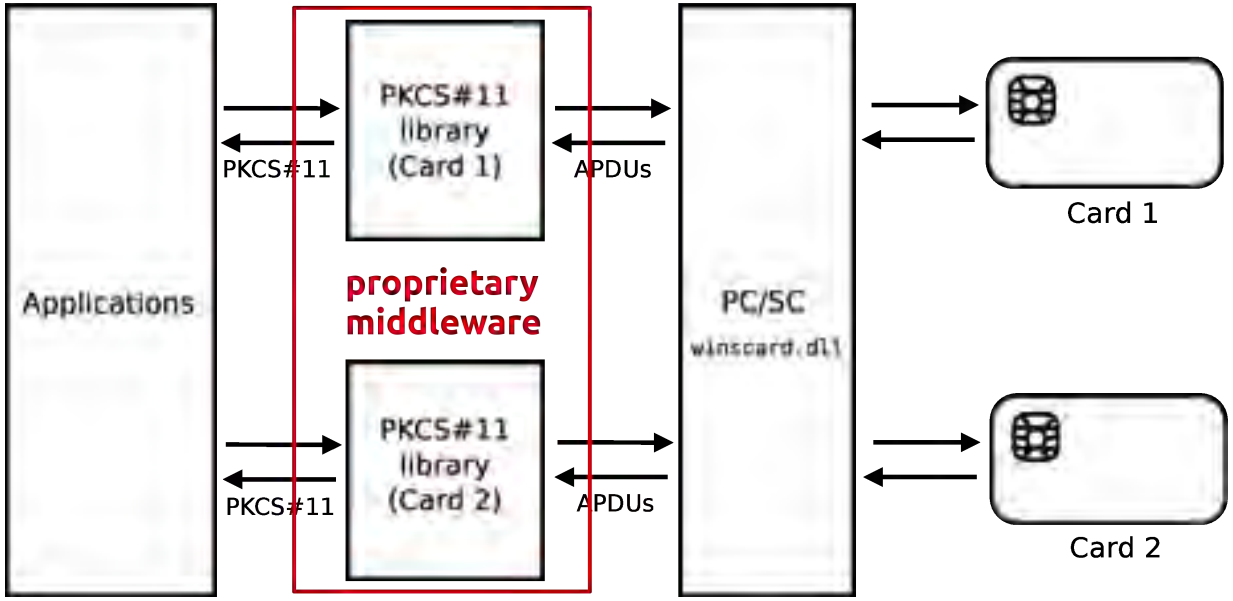
A software layer (middleware) translates PKCS#11 commands into ISO 7816 Application Protocol Data Units (APDUs) [4].



Background

PKCS#11 middleware architecture

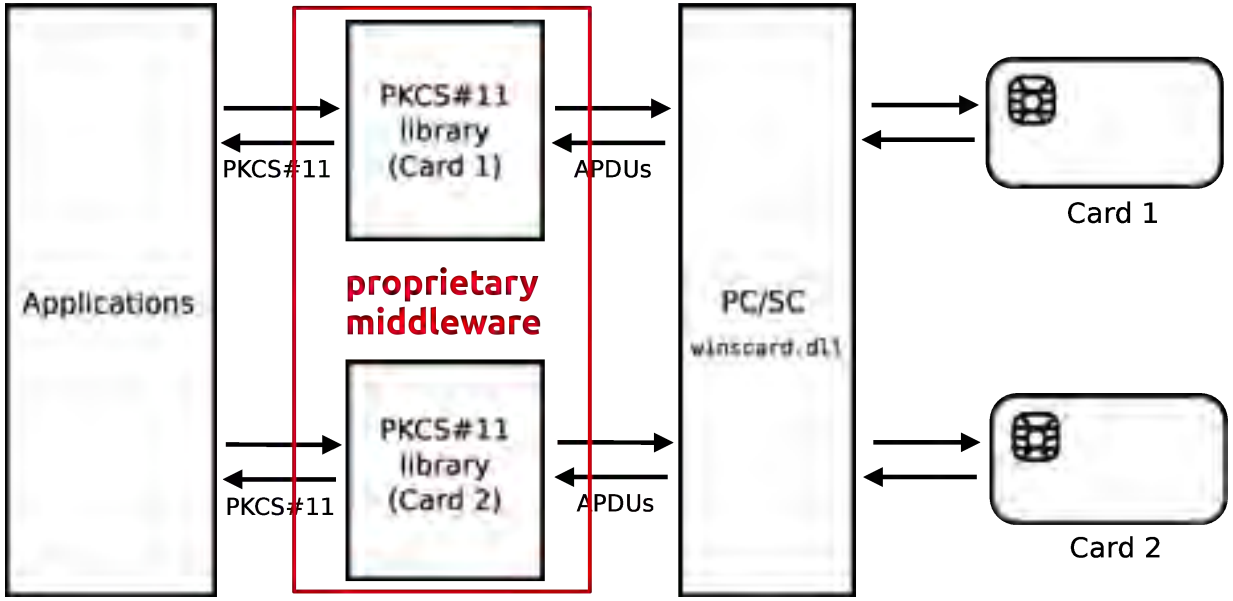
A software layer (middleware) translates PKCS#11 commands into ISO 7816 Application Protocol Data Units (APDUs) [4].



Background

PKCS#11 middleware architecture

A software layer (middleware) translates PKCS#11 commands into ISO 7816 Application Protocol Data Units (APDUs) [4].



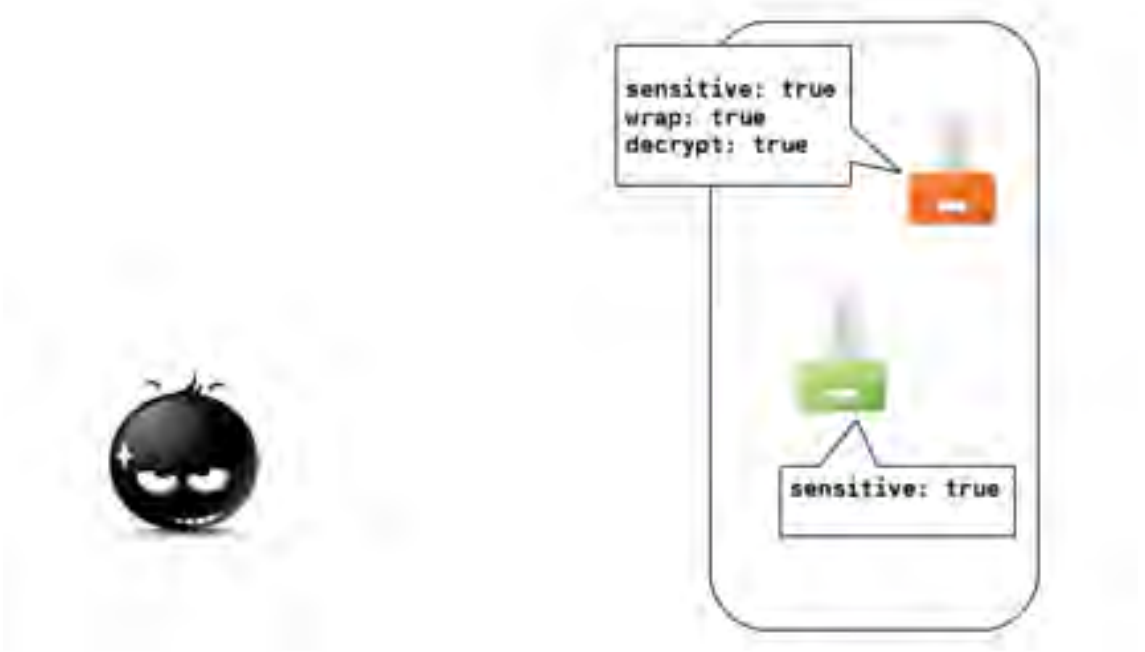
- Known attacks at the PKCS#11-level
- Focus on new attacks at the APDU-level

Background

PKCS#11-level attacks

There exist several well known attacks[1, 2] at the PKCS#11 level. Many of these are key separation attacks, *i.e.* attributes of a key are set so to give a key conflicting roles.

Wrap and decrypt attack:

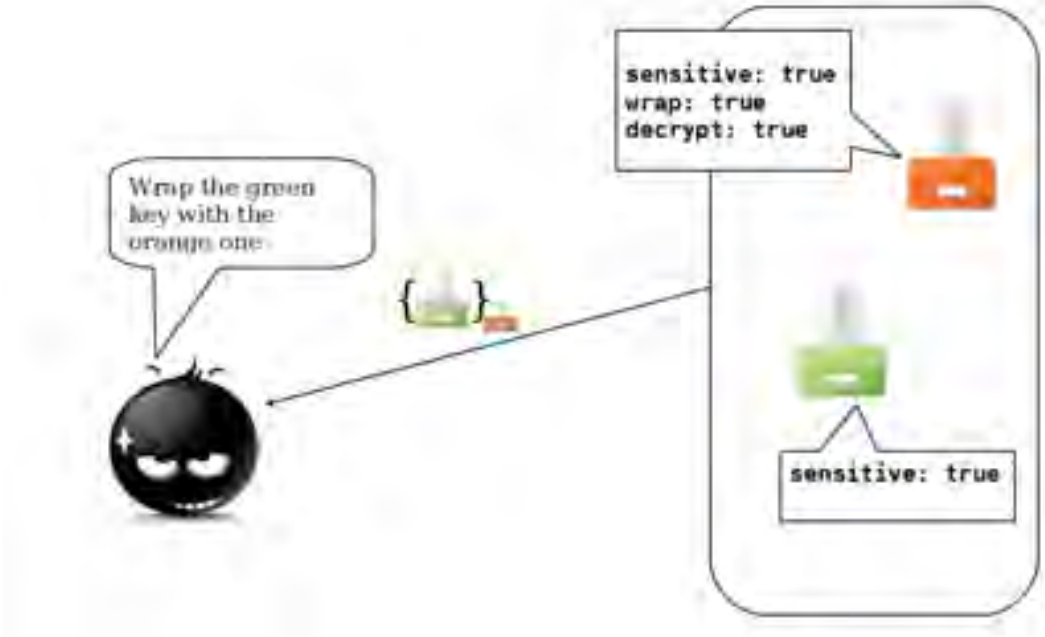


Background

PKCS#11-level attacks

There exist several well known attacks[1, 2] at the PKCS#11 level. Many of these are key separation attacks, *i.e.* attributes of a key are set so to give a key conflicting roles.

Wrap and decrypt attack:

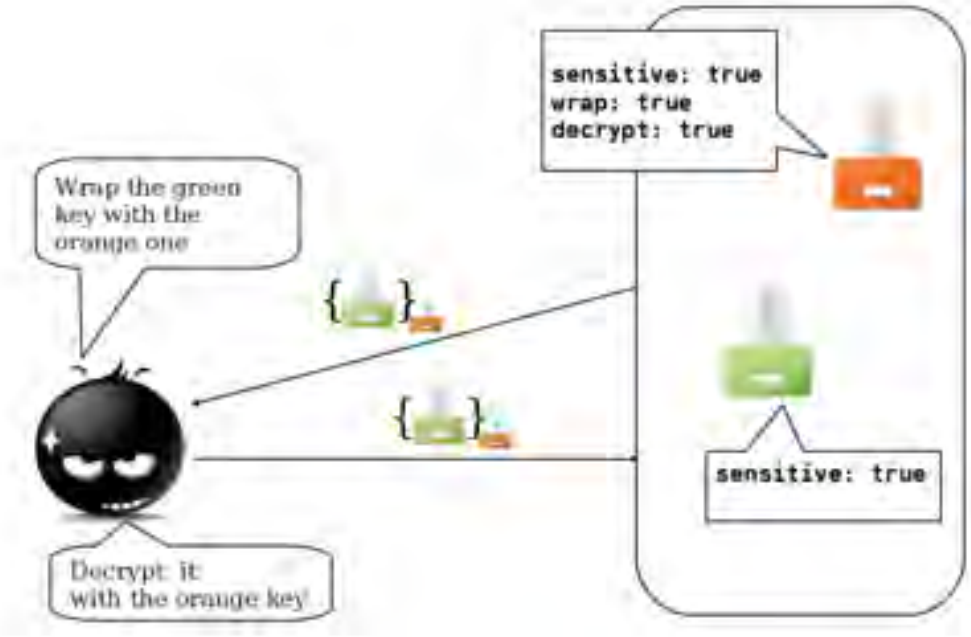


Background

PKCS#11-level attacks

There exist several well known attacks[1, 2] at the PKCS#11 level. Many of these are key separation attacks, *i.e.* attributes of a key are set so to give a key conflicting roles.

Wrap and decrypt attack:

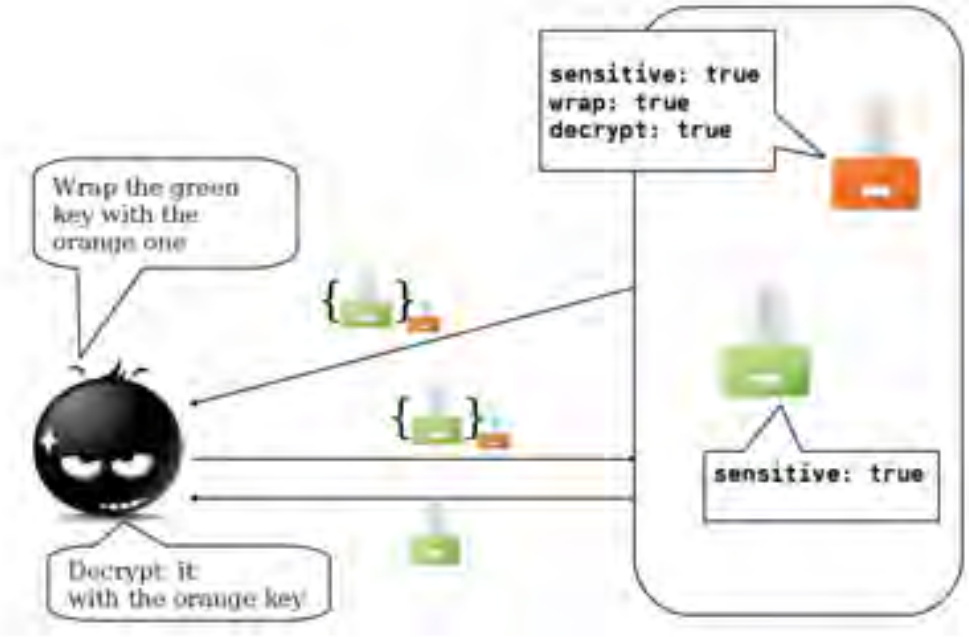


Background

PKCS#11-level attacks

There exist several well known attacks[1, 2] at the PKCS#11 level. Many of these are key separation attacks, *i.e.* attributes of a key are set so to give a key conflicting roles.

Wrap and decrypt attack:



A threat model for PKCS#11 middleware

First threat model for PKCS#11 middleware in literature

Security goals

We focus on these sensitive targets:

- PIN enabling cryptographic operations with the device;
- Cryptographic operations that can be performed independently of the knowledge of the PIN;
- Cryptographic keys leaked in the clear out of the device.

A threat model for PKCS#11 middleware

Typical scenario

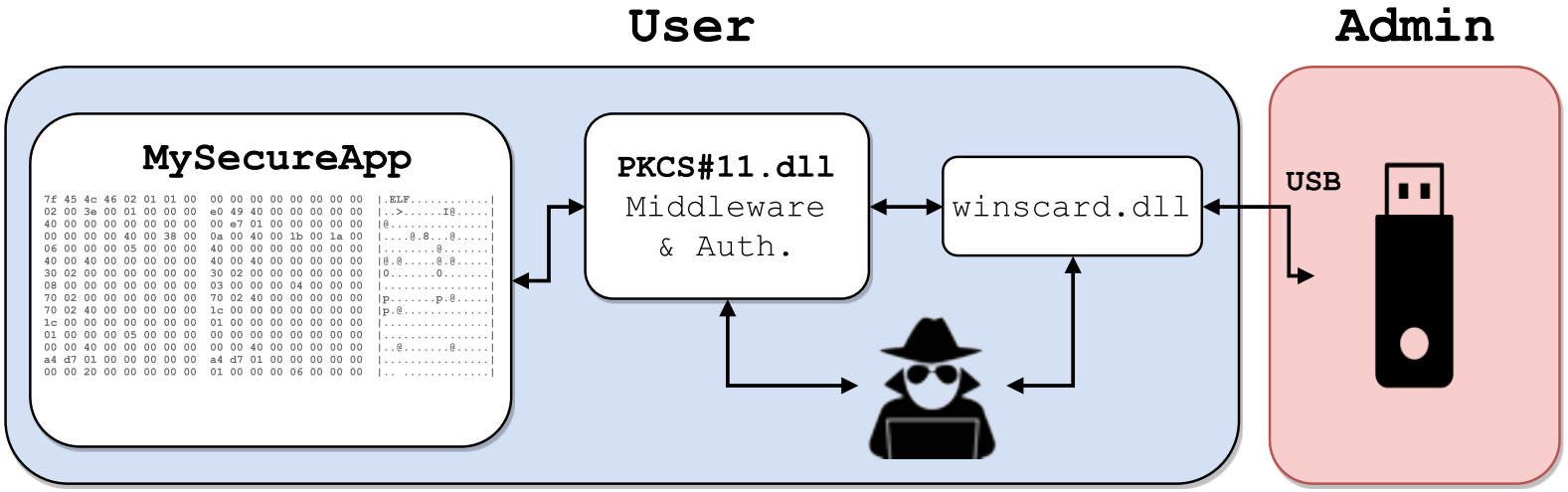
The target token is connected to a desktop/laptop computer in a single-user configuration.

Attacker capabilities

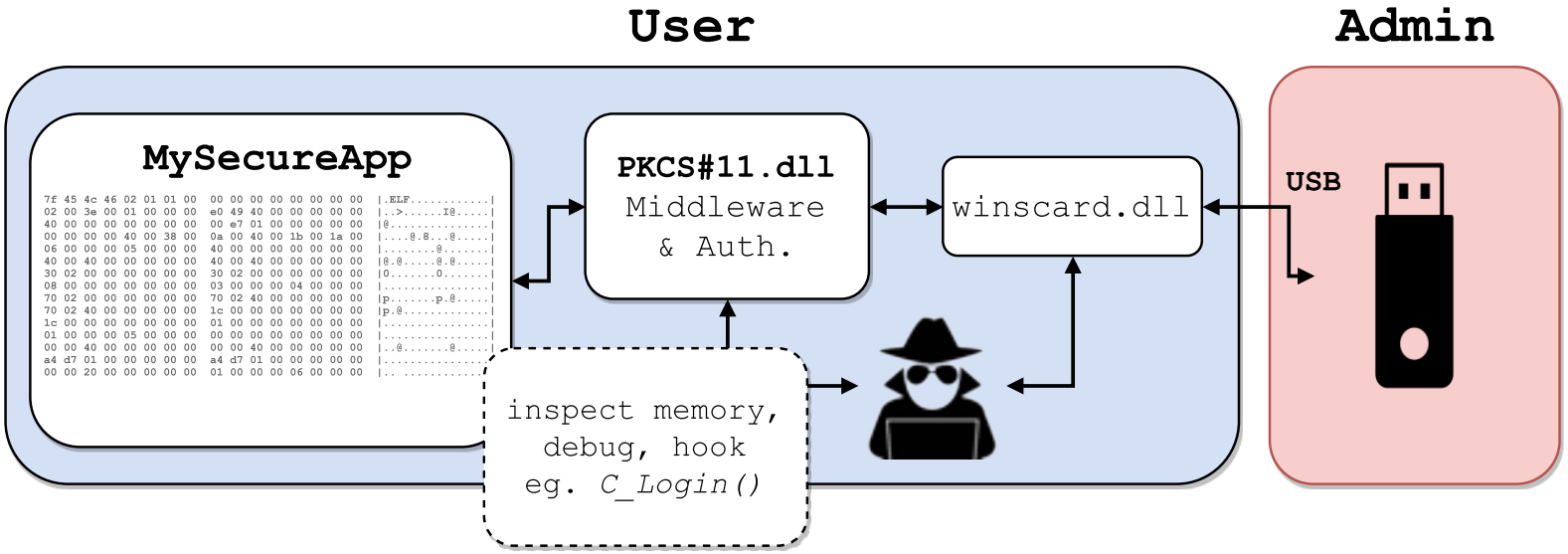
Threat scenarios are classified based on the attacker capabilities:

- Administrator privileges has complete control of the host;
- Physical access can install key-loggers or USB sniffers;
- User privileges has the same privilege level as the regular user.

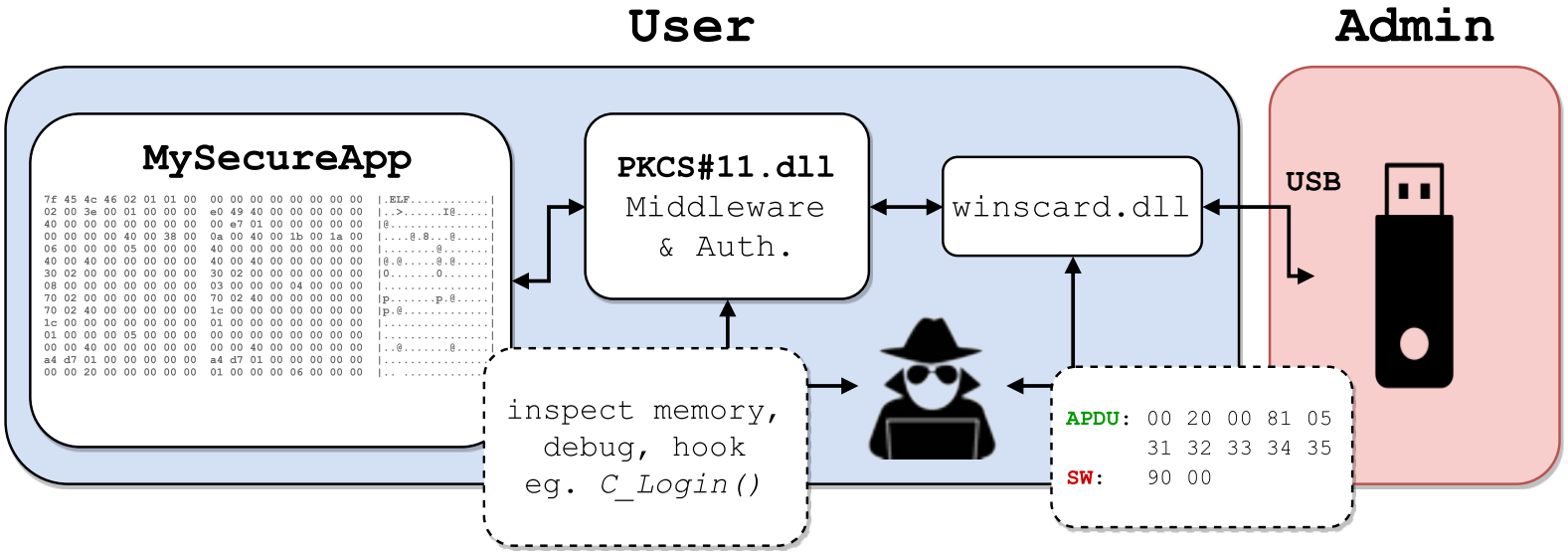
Threat model: monolithic



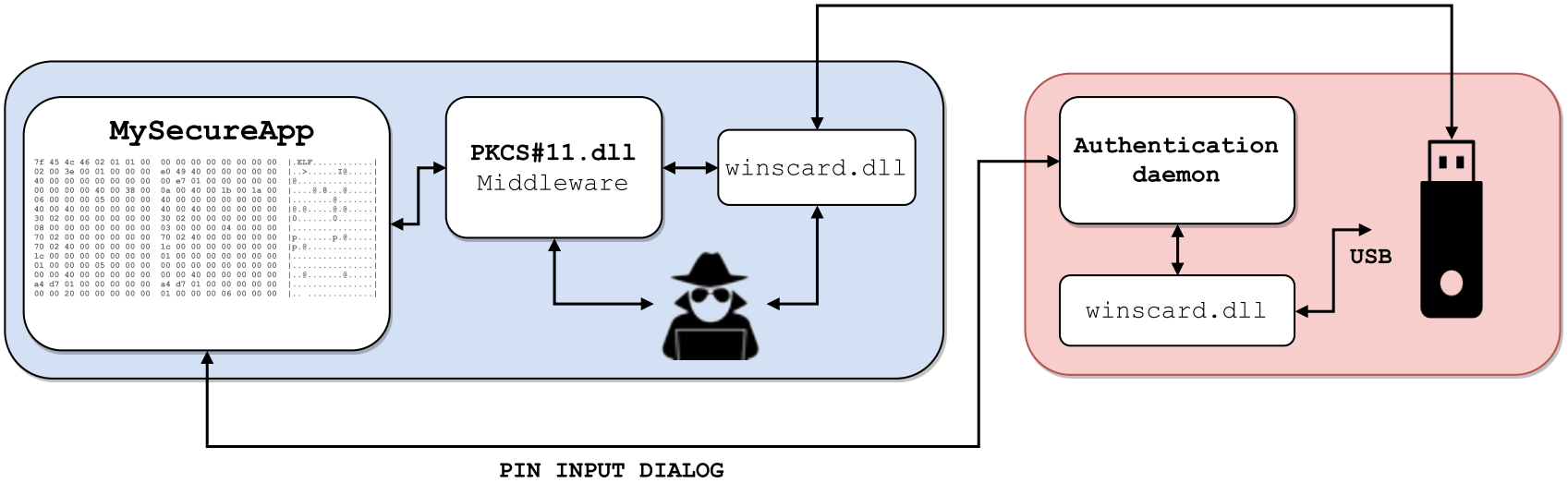
Threat model: monolithic



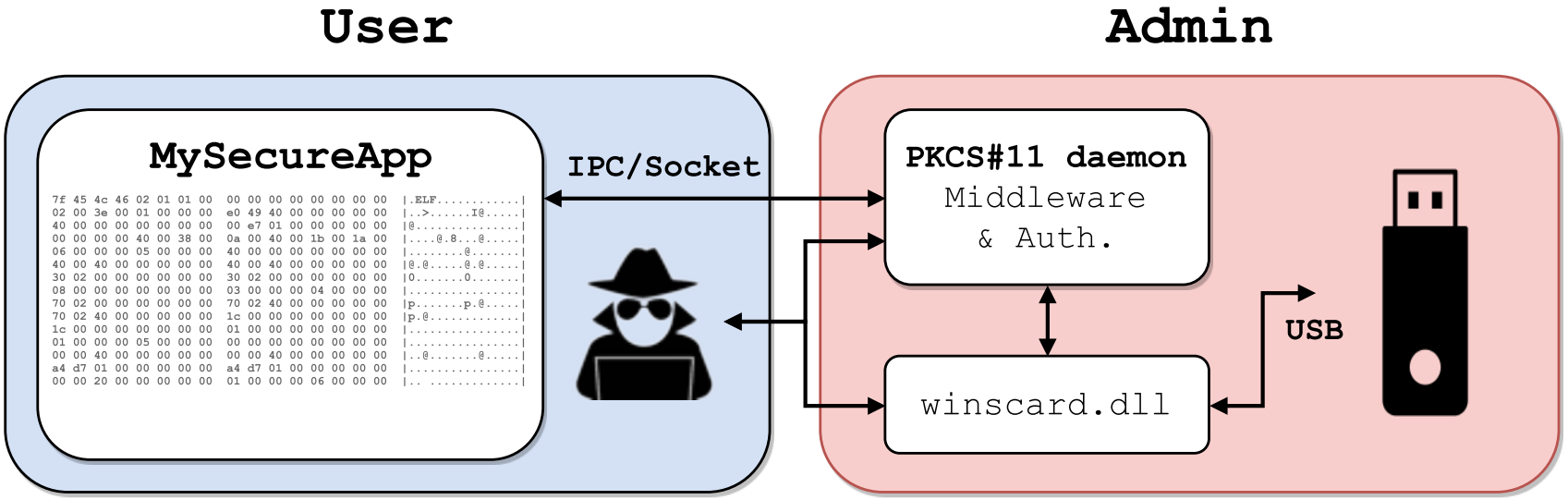
Threat model: monolithic



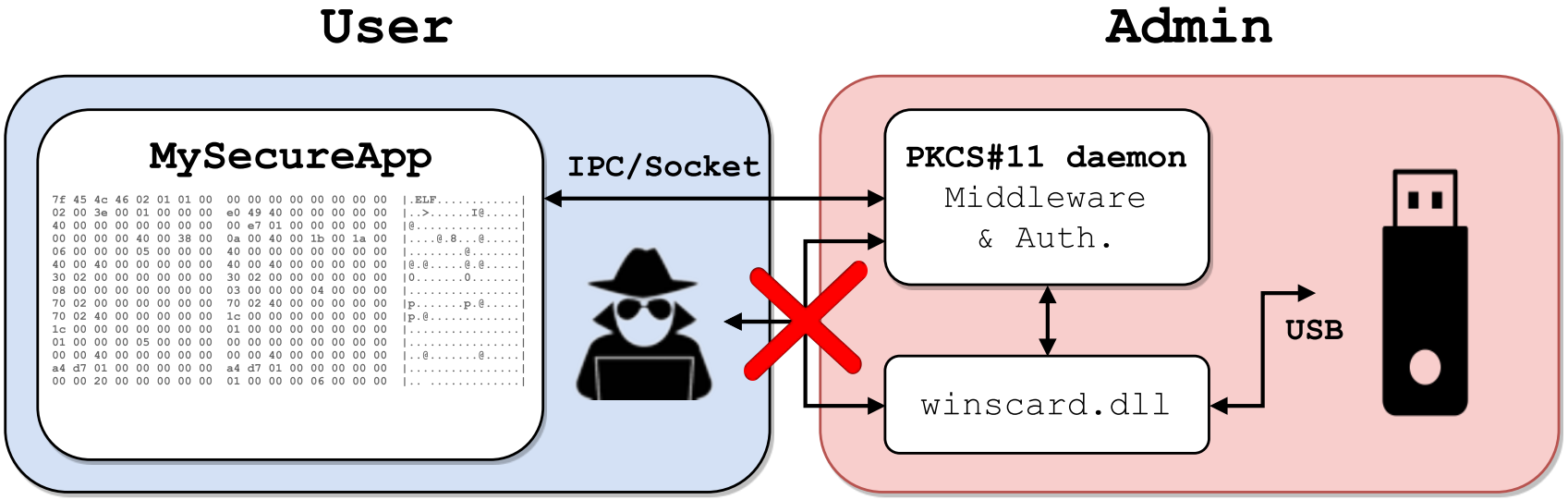
Threat model: separate authentication



Threat model: separate middleware & authentication



Threat model: separate middleware & authentication



Threat model

How these settings affect the attacker's capabilities?

Attacker	Application	Attacker can access		Attacker can exploit			
		PKCS#11	APDU	PIN	PKCS#11	APDU passive	APDU active
Admin	Any	✓	✓	✓	✓	✓	✓
User	Monolithic	✓	✓	✓	✓	✓	✓
	Sep. Auth.	✓	✓	✗	✓[1]	✓	✓[1]
	Sep. Privileges	✓	✗	✓	✓	✗	✗
	Sep. Auth.&Priv.	✓	✗	✗	✓[1]	✗	✗
Physical	Any	✗	✓	✓[2]	✓[1]	✓[3]	✓[1,3]

[1] Requires MITM.
[2] Through a keylogger or a USB sniffer.
[3] Only APDU payloads, cannot access middleware memory.

Threat model

How these settings affect the attacker's capabilities?

Attacker	Application	Attacker can access		Attacker can exploit			
		PKCS#11	APDU	PIN	PKCS#11	APDU passive	APDU active
Admin	Any	✓	✓	✓	✓	✓	✓
User	Monolithic	✓	✓	✓	✓	✓	✓
	Sep. Auth.	✓	✓	✗	✓[1]	✓	✓[1]
	Sep. Privileges	✓	✗	✓	✓	✗	✗
	Sep. Auth.&Priv.	✓	✗	✗	✓[1]	✗	✗
Physical	Any	✗	✓	✓[2]	✓[1]	✓[3]	✓[1,3]

[1] Requires MITM.
[2] Through a keylogger or a USB sniffer.
[3] Only APDU payloads, cannot access middleware memory.

Threat model

How these settings affect the attacker's capabilities?

Attacker	Application	Attacker can access		Attacker can exploit			
		PKCS#11	APDU	PIN	PKCS#11	APDU passive	APDU active
Admin	Any	✓	✓	✓	✓	✓	✓
User	Monolithic	✓	✓	✓	✓	✓	✓
	Sep. Auth.	✓	✓	✗	✓[1]	✓	✓[1]
	Sep. Privileges	✓	✗	✓	✓	✗	✗
	Sep. Auth.&Priv.	✓	✗	✗	✓[1]	✗	✗
Physical	Any	✗	✓	✓[2]	✓[1]	✓[3]	✓[1,3]

[1] Requires MITM.
[2] Through a keylogger or a USB sniffer.
[3] Only APDU payloads, cannot access middleware memory.

APDU-level attacks on real devices

Tested devices

1. Aladdin eToken PRO
2. Athena ASEKey
3. RSA SecurID 800
4. Safesite Classic TPC IS V1
5. Siemens CardOS V4.3b

Vulnerabilities found

- Authentication
- Symmetric keys
- Key attributes handling
- RSA session keys

1)


2)


3)

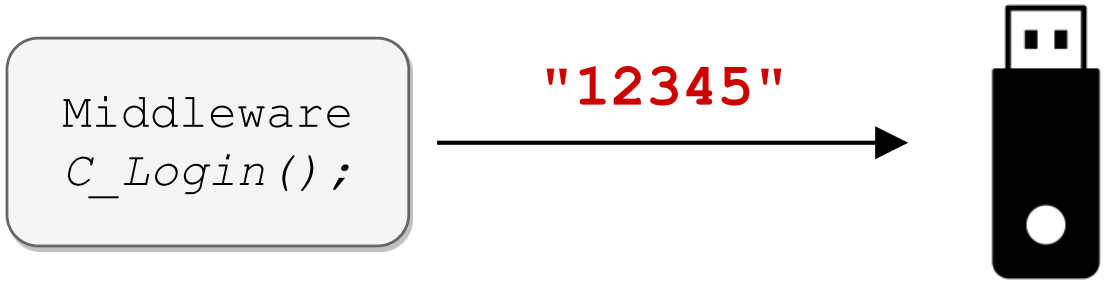

4)


5)


APDU-level attacks: authentication

C_Login() on Siemens CardOS V4.3b:

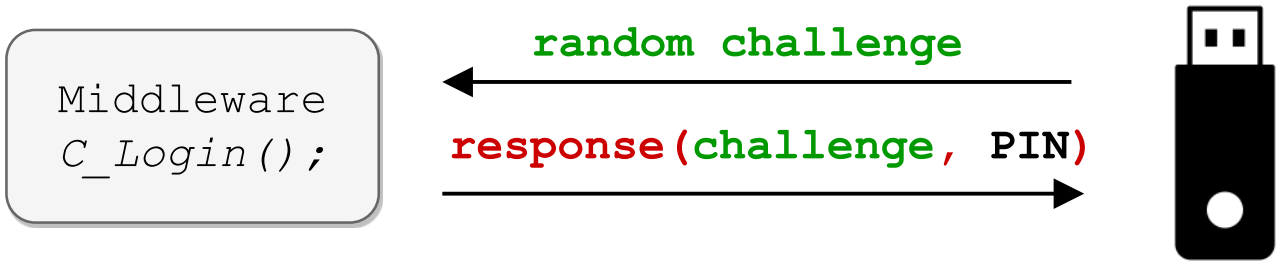
```
# Standard ISO-7816 Select file:
APDU: 00 a4 04 0c 0c a0 00 00 00 63 50 4b 43 53 2d 31 35
SW:   90 00

# Standard ISO-7816 Verify:
APDU: 00 20 00 81 05 31 32 33 34 35
SW:   90 00
```



APDU-level attacks: authentication

C_Login() on Aladdin eToken PRO:

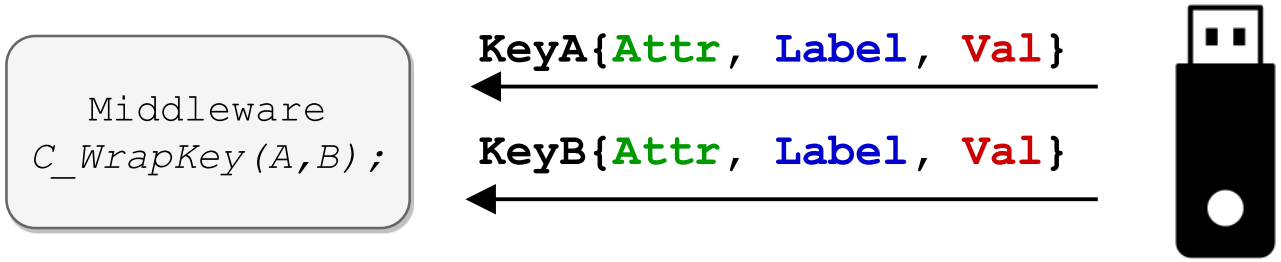
```
# Custom Get challenge:
APDU: 80 17 00 00 08
SW:   DF 89 61 34 62 05 13 36  90 00

# Custom External authenticate:
APDU: 80 11 00 11 0A 10 08  64 D5 97 15 4A 44 EB 23
SW:   90 00
```



APDU-level attacks: symmetric keys

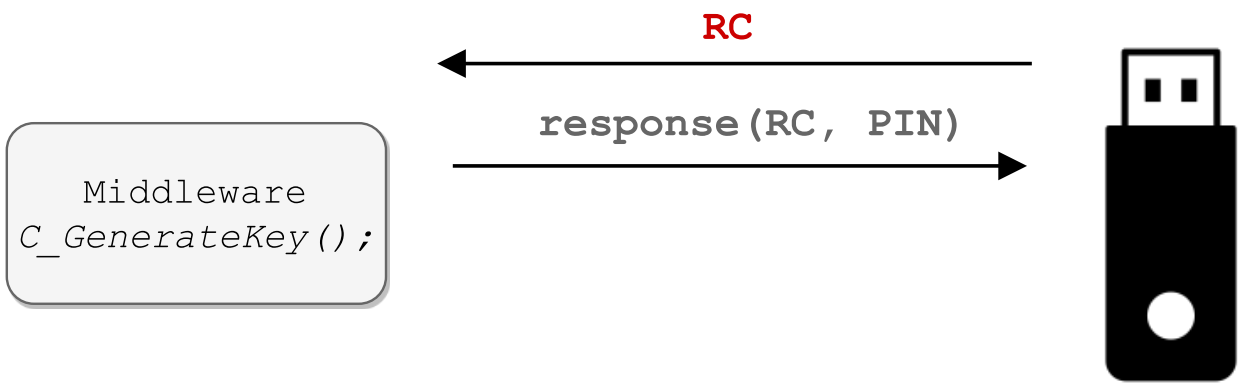
C_WrapKey() on Aladdin eToken PRO:

```
# Fetch the key
APDU: 80 18 00 00 04 0E 02 00 00 18
SW:   17 3F FF FF FF FF 01 08 3F 44 5F C4 EB 76 F1 86
      06 64 65 73 6B 65 79 00 90 00
```



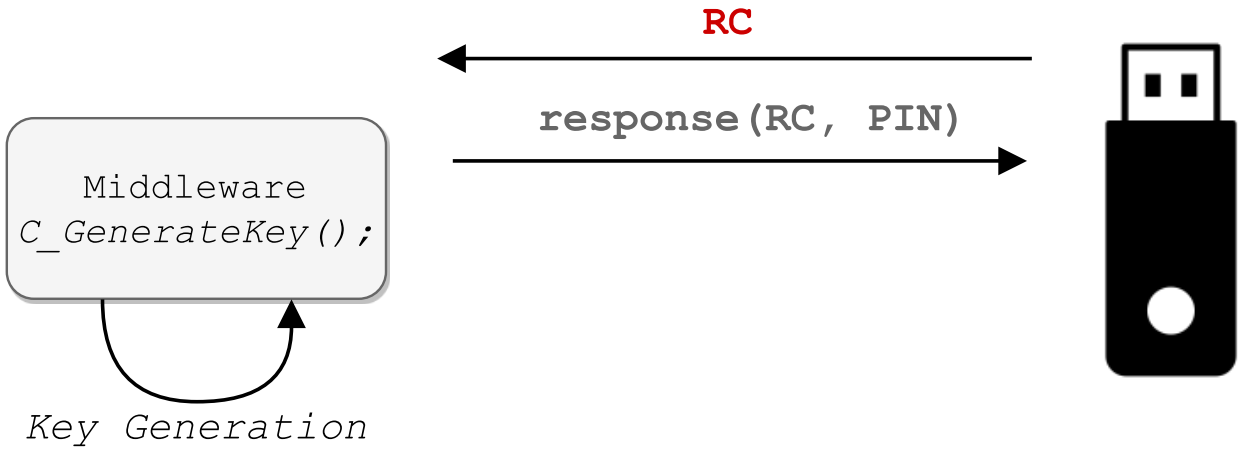
C_GenerateKey() (DES key) on Athena ASEKey:

```
# Get challenge (Standard ISO-7816):
APDU: 00 84 00 00 00 00 08
SW:   b7 c8 14 4b 4e 5f e6 3e 90 00
# .... (omitted) ....
# Get an RSA modulus
APDU: 80 14 02 91 00 00 00
SW:   79 23 57 33 9a be 2a dd ba ae 2e 09 4c d0 3d 57
      8b d0 07 e4 cb ..(omitted).. 30 c3 e8 cf 90 00
# Send the encrypted key to the token
APDU: 80 24 00 80 00 00 a0 20 5b f1 f9 cd 67 c8 3d e0
      cf 9b 1b c7 ad ..(omitted).. a7 f6 4a 97 22 a0
SW:   90 00
```



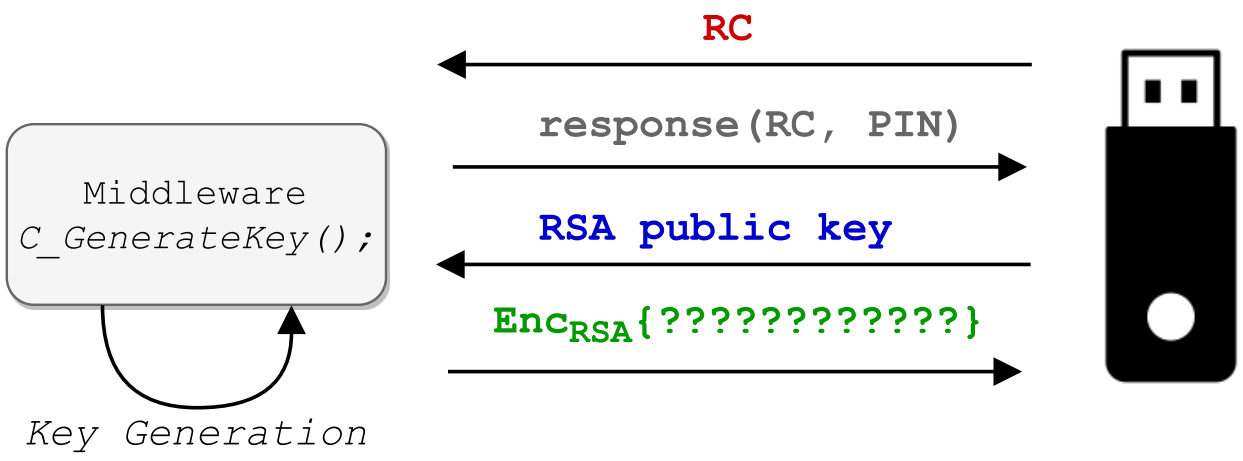
C_GenerateKey() (DES key) on Athena ASEKey:

```
# Get challenge (Standard ISO-7816):
APDU: 00 84 00 00 00 00 08
SW:   b7 c8 14 4b 4e 5f e6 3e 90 00
# .... (omitted) ....
# Get an RSA modulus
APDU: 80 14 02 91 00 00 00
SW:   79 23 57 33 9a be 2a dd ba ae 2e 09 4c d0 3d 57
      8b d0 07 e4 cb ..(omitted).. 30 c3 e8 cf 90 00
# Send the encrypted key to the token
APDU: 80 24 00 80 00 00 a0 20 5b f1 f9 cd 67 c8 3d e0
      cf 9b 1b c7 ad ..(omitted).. a7 f6 4a 97 22 a0
SW:   90 00
```



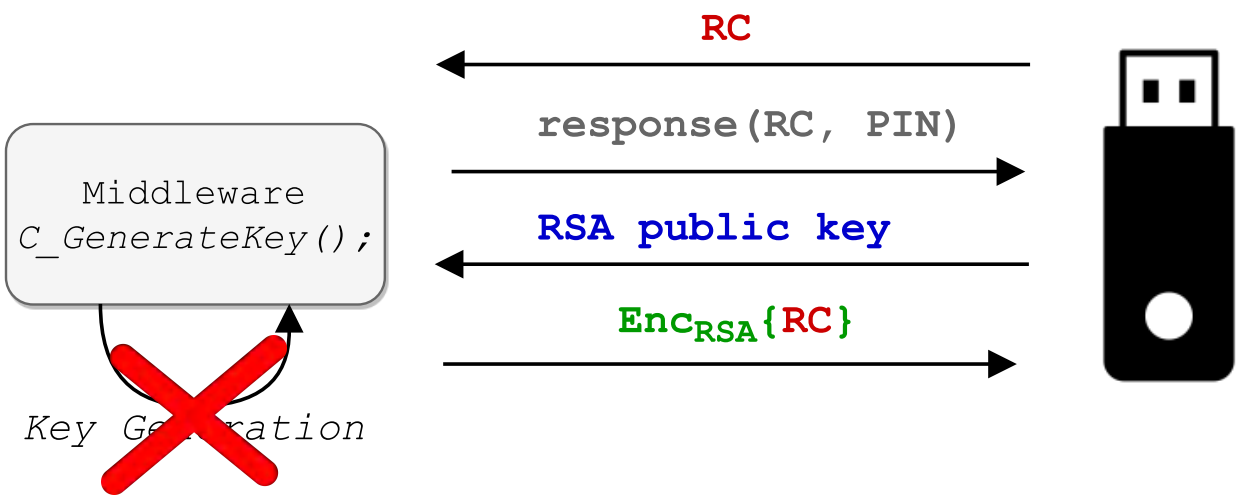
C_GenerateKey() (DES key) on Athena ASEKey:

```
# Get challenge (Standard ISO-7816):
APDU: 00 84 00 00 00 00 08
SW:   b7 c8 14 4b 4e 5f e6 3e 90 00
# .... (omitted) ....
# Get an RSA modulus
APDU: 80 14 02 91 00 00 00
SW:   79 23 57 33 9a be 2a dd ba ae 2e 09 4c d0 3d 57
      8b d0 07 e4 cb ..(omitted).. 30 c3 e8 cf 90 00
# Send the encrypted key to the token
APDU: 80 24 00 80 00 00 a0 20 5b f1 f9 cd 67 c8 3d e0
      cf 9b 1b c7 ad ..(omitted).. a7 f6 4a 97 22 a0
SW:   90 00
```



C_GenerateKey() (DES key) on Athena ASEKey:

```
# Get challenge (Standard ISO-7816):
APDU: 00 84 00 00 00 00 08
SW:   b7 c8 14 4b 4e 5f e6 3e 90 00
# .... (omitted) ....
# Get an RSA modulus
APDU: 80 14 02 91 00 00 00
SW:   79 23 57 33 9a be 2a dd ba ae 2e 09 4c d0 3d 57
      8b d0 07 e4 cb ..(omitted).. 30 c3 e8 cf 90 00
# Send the encrypted key to the token
APDU: 80 24 00 80 00 00 a0 20 5b f1 f9 cd 67 c8 3d e0
      cf 9b 1b c7 ad ..(omitted).. a7 f6 4a 97 22 a0
SW:   90 00
```



APDU-level attacks: key attributes

A key with **CKA_SIGN** attribute set to `CK_FALSE` cannot perform signature operations.

Bypassing attribute values

1. take a private RSA key with `CKA_SIGN` false;
2. verify that it cannot sign a message via the `PKCS#11` API;
3. perform the sign operation manually, via `APDU`.

APDU-level attacks: summary

Token	Auth.	Sensitive symmetric keys		Bypassing attribute values	RSA session keys	
		PKCS#11[1]	APDU		PKCS#11[1]	APDU
eToken PRO	✓[2]	✓	✓	✓	✗	✓[4]
ASEKey	✓[2]	✗	✓[3]	✓	✗	✗
SecurID	✓[2]	✓[5]	✓	✓	✗	✗
Safesite Classic	✓	✗	✗	✓	✗	✗
Siemens CardOS	✓	✗	✓[4]	✓	✗	✗

[1] PKCS#11-level attacks discovered in [1], for comparison.
[2] Requires reverse engineering of the authentication algorithm and bruteforcing.
[3] Leakage occurs only during generation.
[4] Requires access to middleware memory.
[5] Possible for RSA Authentication Client version $< 3.5.3$.

APDU-level attacks: summary

Token	Auth.	Sensitive symmetric keys		Bypassing attribute values	RSA session keys	
		PKCS#11[1]	APDU		PKCS#11[1]	APDU
eToken PRO	✓[2]	✓	✓	✓	✗	✓[4]
ASEKey	✓[2]	✗	✓[3]	✓	✗	✗
SecurID	✓[2]	✓[5]	✓	✓	✗	✗
Safesite Classic	✓	✗	✗	✓	✗	✗
Siemens CardOS	✓	✗	✓[4]	✓	✗	✗

[1] PKCS#11-level attacks discovered in [1], for comparison.
[2] Requires reverse engineering of the authentication algorithm and bruteforcing.
[3] Leakage occurs only during generation.
[4] Requires access to middleware memory.
[5] Possible for RSA Authentication Client version $< 3.5.3$.

APDU-level attacks: summary

Token	Auth.	Sensitive symmetric keys		Bypassing attribute values	RSA session keys	
		PKCS#11[1]	APDU		PKCS#11[1]	APDU
eToken PRO	✓[2]	✓	✓	✓	✗	✓[4]
ASEKey	✓[2]	✗	✓[3]	✓	✗	✗
SecurID	✓[2]	✓[5]	✓	✓	✗	✗
Safesite Classic	✓	✗	✗	✓	✗	✗
Siemens CardOS	✓	✗	✓[4]	✓	✗	✗

[1] PKCS#11-level attacks discovered in [1], for comparison.
[2] Requires reverse engineering of the authentication algorithm and bruteforcing.
[3] Leakage occurs only during generation.
[4] Requires access to middleware memory.
[5] Possible for RSA Authentication Client version $< 3.5.3$.

APDU-level attacks: summary

Token	Auth.	Sensitive symmetric keys		Bypassing attribute values	RSA session keys	
		PKCS#11[1]	APDU		PKCS#11[1]	APDU
eToken PRO	✓[2]	✓	✓	✓	✗	✓[4]
ASEKey	✓[2]	✗	✓[3]	✓	✗	✗
SecurID	✓[2]	✓[5]	✓	✓	✗	✗
Safesite Classic	✓	✗	✗	✓	✗	✗
Siemens CardOS	✓	✗	✓[4]	✓	✗	✗

[1] PKCS#11-level attacks discovered in [1], for comparison.
[2] Requires reverse engineering of the authentication algorithm and bruteforcing.
[3] Leakage occurs only during generation.
[4] Requires access to middleware memory.
[5] Possible for RSA Authentication Client version $< 3.5.3$.

APDU-level attacks: summary

Token	Auth.	Sensitive symmetric keys		Bypassing attribute values	RSA session keys	
		PKCS#11[1]	APDU		PKCS#11[1]	APDU
eToken PRO	✓[2]	✓	✓	✓	✗	✓[4]
ASEKey	✓[2]	✗	✓[3]	✓	✗	✗
SecurID	✓[2]	✓[5]	✓	✓	✗	✗
Safesite Classic	✓	✗	✗	✓	✗	✗
Siemens CardOS	✓	✗	✓[4]	✓	✗	✗

[1] PKCS#11-level attacks discovered in [1], for comparison.
[2] Requires reverse engineering of the authentication algorithm and bruteforcing.
[3] Leakage occurs only during generation.
[4] Requires access to middleware memory.
[5] Possible for RSA Authentication Client version $< 3.5.3$.

Fixes and mitigations

Compliant PKCS#11 devices should implement all the cryptographic operations inside the hardware

Fixes

- Hardware/firmware redesign
- Separate authentication
- Higher privileges middleware

Fixes and mitigations

Compliant PKCS#11 devices should implement all the cryptographic operations inside the hardware

Fixes

- Hardware/firmware redesign → costly
- Separate authentication
- Higher privileges middleware

Fixes and mitigations

Compliant PKCS#11 devices should implement all the cryptographic operations inside the hardware

Fixes

- Hardware/firmware redesign → costly
- Separate authentication → not backward compatible
- Higher privileges middleware

Fixes and mitigations

Compliant PKCS#11 devices should implement all the cryptographic operations inside the hardware

Fixes

- Hardware/firmware redesign → costly
- Separate authentication → not backward compatible
- Higher privileges middleware → transparent

Fixes and mitigations

Compliant PKCS#11 devices should implement all the cryptographic operations inside the hardware

Fixes

- Hardware/firmware redesign → costly
- Separate authentication → not backward compatible
- Higher privileges middleware → transparent

Mitigation: OTP

$C\_Login(OTP + PIN)$

Conclusion

Contributions

- We introduced a new threat model for PKCS#11 middleware
- We found new, unpublished APDU-level attacks on commercially available tokens and smartcards
- We provided a security analysis of the vulnerabilities with respect to the threat model

Future works

Wish list...

- Test newer devices
- Encrypted `APDU`s
- PIN bruteforcing using card emulation
- Open hardware-firmware-middleware reference implementation

Thank you!

no tokens were harmed during the making of this paper

Want more?

Really?

https://secgroup.dais.unive.it/projects/apduattacks/

- Summary of the paper as a blog post
- Official answers from manufacturers
- Previous / future works [1, 3]

References

- Bortolozzo, M., Centenaro, M., Focardi, R., Steel, G.: Attacking and fixing PKCS#11 security tokens. In: Proceedings of the 17th ACM Conference on Computer and Communications Security (CCS'10). pp. 260–269. ACM (2010)
- Clulow, J.: On the security of PKCS#11. In: 5th International Workshop on Cryptographic Hardware and Embedded Systems (CHES'03). LNCS, vol. 2779, pp. 411–425. Springer (2003)
- Gkaniatsou, A., McNeill, F., Bundy, A., Steel, G., Focardi, R., Bozzato, C.: Getting to know your card: Reverse-engineering the smart-card application protocol data unit. In: Proceedings of the 31st Annual Computer Security Applications Conference, Los Angeles, CA, USA, December 7-11, 2015. pp. 441–450 (2015)
- ISO/IEC 7816-4: Identification cards – Integrated circuit cards – Part 4: Organization, security and commands for interchange (2013)

Backup slides

Related works

PKCS#11

- First attacks on PKCS#11 by Clulow
- General analysis tools for PKCS#11
- Generalization of the model and automatic reverse engineering tool

Low level

- No previous APDU-level attacks and threat models for PKCS#11 devices
- APDU buffer compromised in Java Cards
- MITM attack for payments w/o needing PIN
- Automated method to reverse engineer PKCS#11-APDU mapping