# Getting to know your card: Reverse-Engineering the Smart-Card Application Protocol Data Unit for PKCS#11 Functions

*Andriana Gkaniatsou*[1], Fiona McNeill[2],
Alan Bundy[1], Graham Steel[3]
Riccardo Focardi[4], Claudio Bozzato[4]
[1]University of Edinburgh[2] Heriot-Watt [3]Cryptosense
[4]Ca'Foscari

# Smart-cards

- *secure, trusted, tamper-resistant*
- identification, authentication, data storage and application processing
- financial, communication, security and data management purposes

# Smart-cards

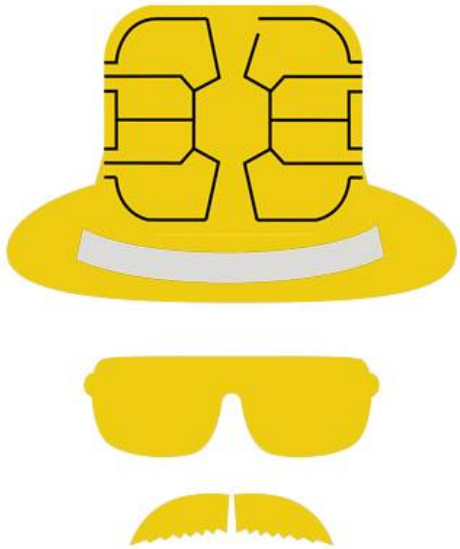
- *secure, trusted, tamper-resistant*
- identification, authentication, data storage and application processing
- financial, communication, security and data management purposes
- *third-party communication*

# Smart-cards

- *secure, trusted, tamper-resistant*
- identification, authentication, data storage and application processing
- financial, communication, security and data management purposes
- *third-party communication*
- *black-box*

# Smart-cards

- *secure, trusted, tamper-resistant*
- identification, authentication, data storage and application processing
- financial, communication, security and data management purposes
- *third-party communication*
- *black-box*



is your card breaking bad?

# Cryptographic protocols

## RSA PKCS# 11 Cryptographic Token Interface Standard

- functions key management, signing, encryption, decryption *etc.*
- ensure sensitive data remain secure

## API-Level Attacks

E.g., Clulow, J., On the security of PKCS# 11. CHES 2003
Bortolozzo, M., Centenaro, M., Focardi, R., & Steel, G. Attacking and fixing PKCS# 11 security tokens. CCS 2010

# Cryptographic protocols

## RSA PKCS# 11 Cryptographic Token Interface Standard

- functions key management, signing, encryption, decryption *etc.*
- ensure sensitive data remain secure

## API-Level Attacks

E.g., Clulow, J., On the security of PKCS# 11. CHES 2003
Bortolozzo, M., Centenaro, M., Focardi, R., & Steel, G. Attacking and fixing PKCS# 11 security tokens. CCS 2010

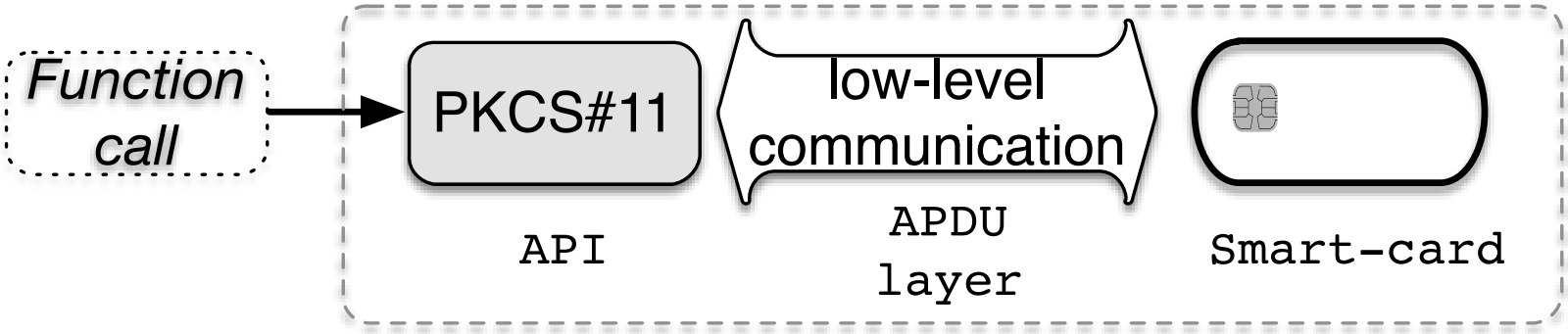
## PKCS#11 Low-level Implementation

**has been kept in the dark**

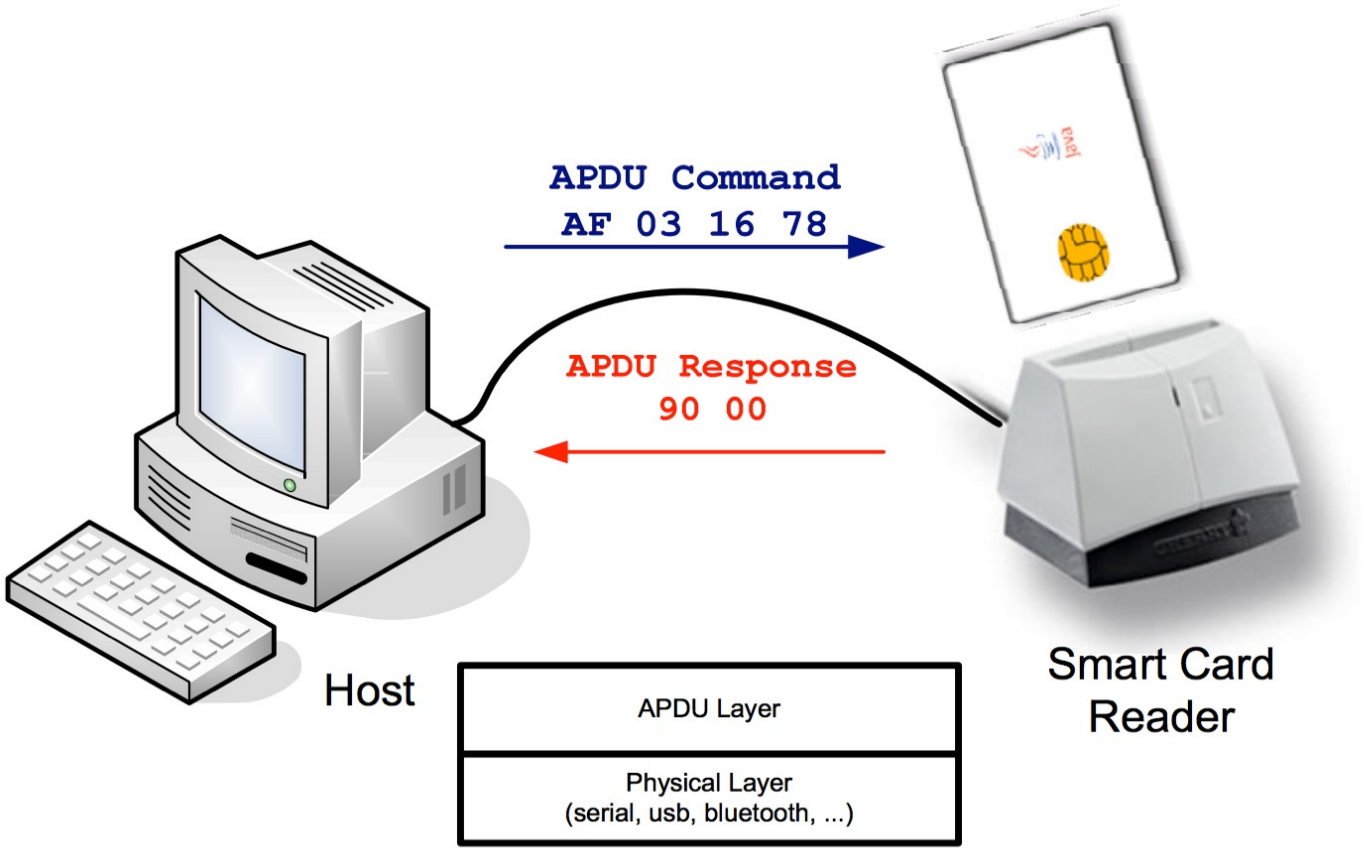
# Smart-card communication

## Smart-card Communication

**How is PKCS#11 implemented at the lowest-level communication? Is it secure?**

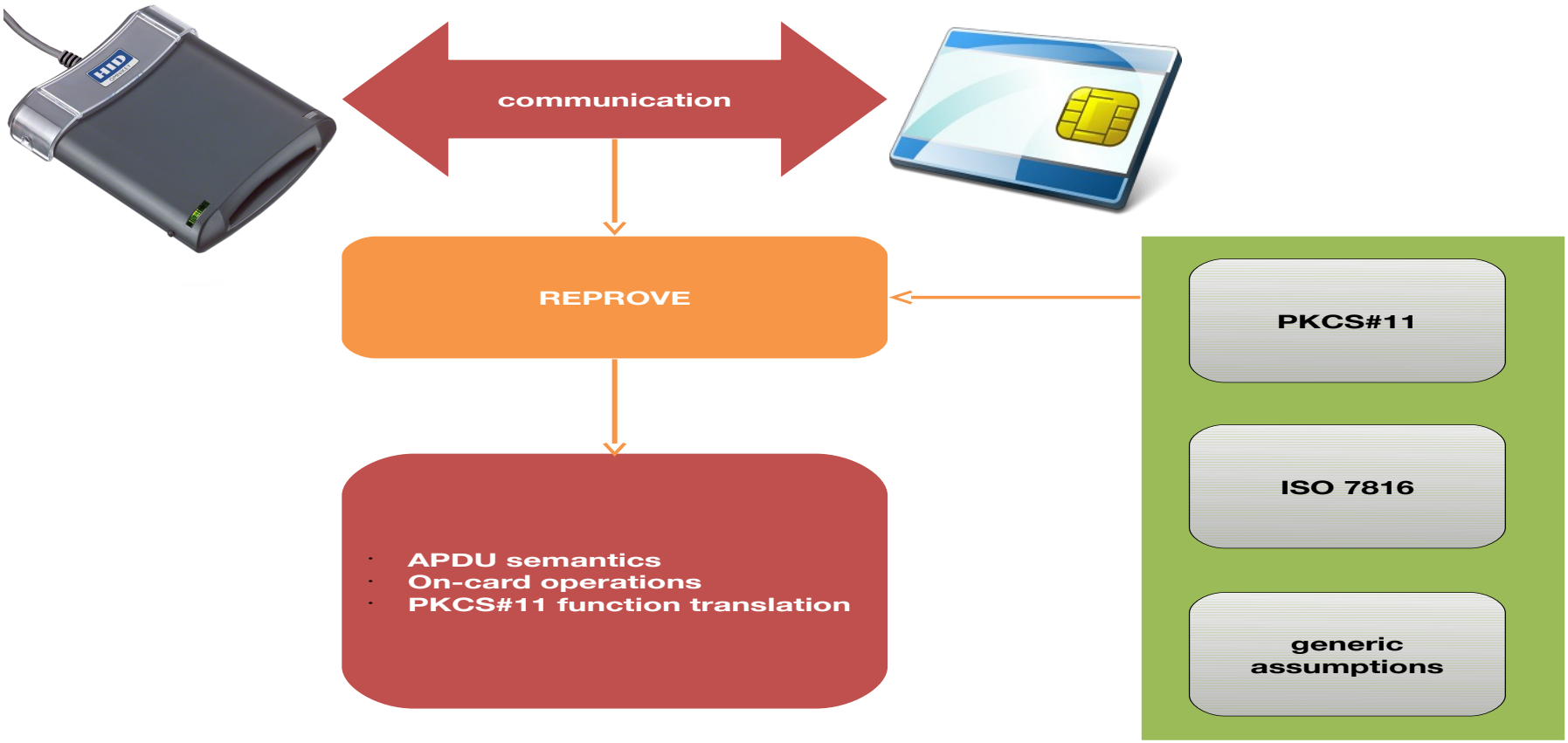


# Smart-card communication



# The REPROVE system

REPROVE reverse-engineering system: **no API access - no card access - implementation indepedent**



# ISO/IEC 7816

Defines the communication layer between the card and the reader:15 Parts

- Part 4: Organisation, security and commands for interchange
- Part 8: Commands for security operations
- Part 9: Commands for card management.

# APDU command structure

>00 a4 08 0c 04 50154400 01

<08 9000

# APDU command structure

>00 a4 08 0c 04 50154400 01

- Cla: Instruction Class

<08 9000

# APDU command structure

>00 a4 08 0c 04 50154400 01

- Cla: Instruction Class
- Ins: Instruction Code

<08 9000

# APDU command structure

>00 a4 08 0c 04 50154400 01

- Cla: Instruction Class
- Ins: Instruction Code
- P1-P2: Instruction Parameters

<08 9000

# APDU command structure

>00 a4 08 0c 04 50154400 01

- Cla: Instruction Class
- Ins: Instruction Code
- P1-P2: Instruction Parameters
- Lc: Length of sent data
- D: Sent data

<08 9000

# APDU command structure

>00 a4 08 0c 04 50154400 01

- Cla: Instruction Class
- Ins: Instruction Code
- P1-P2: Instruction Parameters
- Lc: Length of sent data
- D: Sent data
- Le: Length of expected data

<08 9000

# APDU command structure

>00 a4 08 0c 04 50154400 01

- Cla: Instruction Class
- Ins: Instruction Code
- P1-P2: Instruction Parameters
- Lc: Length of sent data
- D: Sent data
- Le: Length of expected data

<08 9000

- D: Response data

# APDU command structure

>00 a4 08 0c 04 50154400 01

- Cla: Instruction Class
- Ins: Instruction Code
- P1-P2: Instruction Parameters
- Lc: Length of sent data
- D: Sent data
- Le: Length of expected data

<08 9000

- D: Response data
- SW1-SW2: Command processing status

# APDU command structure

>00 a4 08 0c 04 50154400 01

- Cla: Instruction Class
- Ins: Instruction Code
- P1-P2: Instruction Parameters
- Lc: Length of sent data
- D: Sent data
- Le: Length of expected data

<08 9000

- D: Response data
- SW1-SW2: Command processing status

>80 21 08 0c 04 50154400 01

# APDU command structure

>00 a4 08 0c 04 50154400 01

- Cla: Instruction Class
- Ins: Instruction Code
- P1-P2: Instruction Parameters
- Lc: Length of sent data
- D: Sent data
- Le: Length of expected data

<08 9000

- D: Response data
- SW1-SW2: Command processing status

>80 21 08 0c 04 50154400 01

**Analysis Challenge**

How can we infer the semantics of the proprietary command?
*e.g.*, 21↦a4 ?

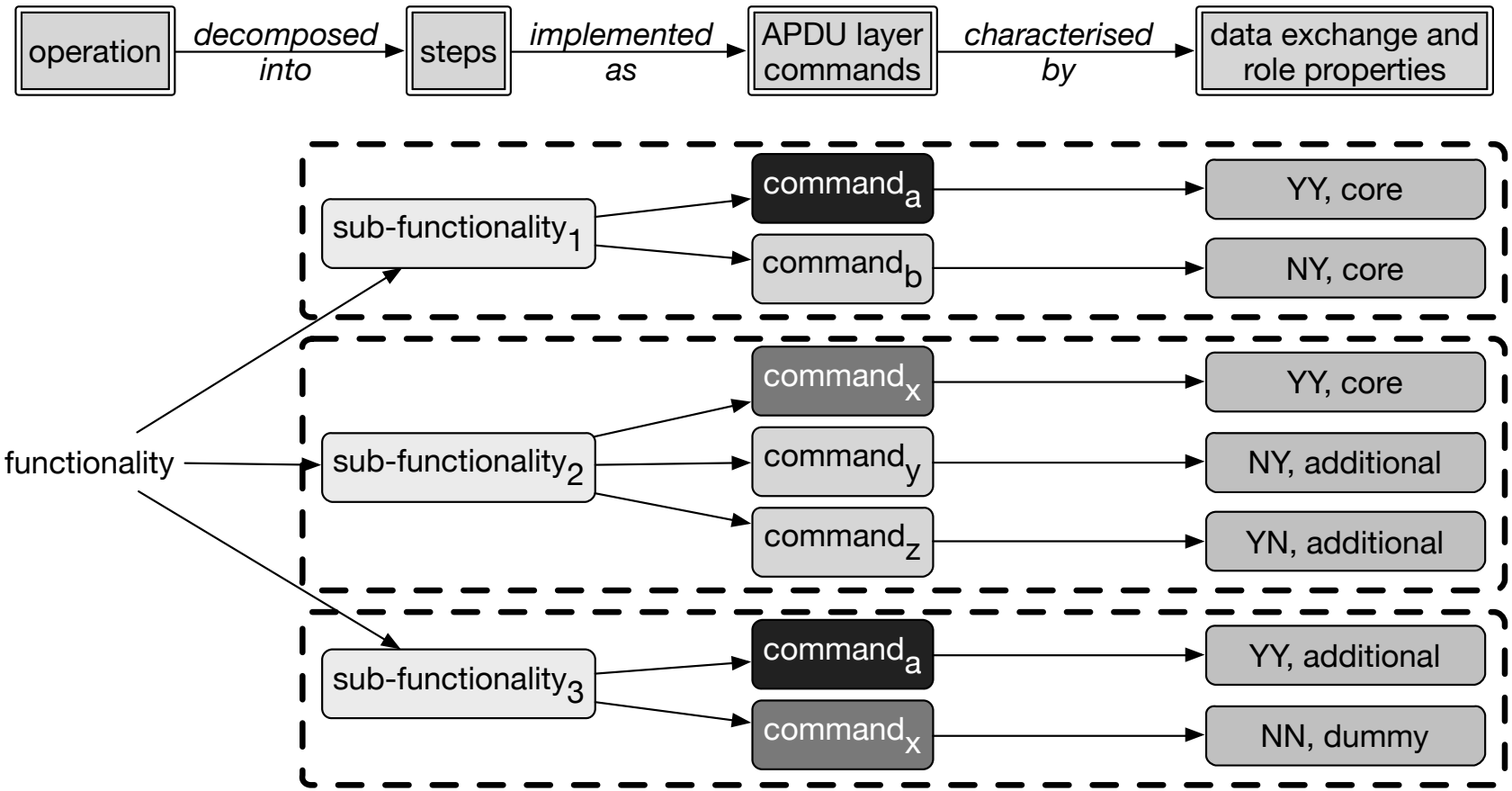
# Methodology

- ISO 7816 models
- Command precondition models
- Command categorization
- Card operations models
- Patterns
- Hierarchy of card operations
- PKCS#11 functions models: C_login, C_generateKey, C_sign, C_findObjectsInit, C_findObjects, C_getAttributeValue, C_setAttributeValue, C_wrapKey, C_encrypt, C_unwrapKey

## Inference Problem

Given a set of models derive the meaning of the actual implementation.

# APDU modelling

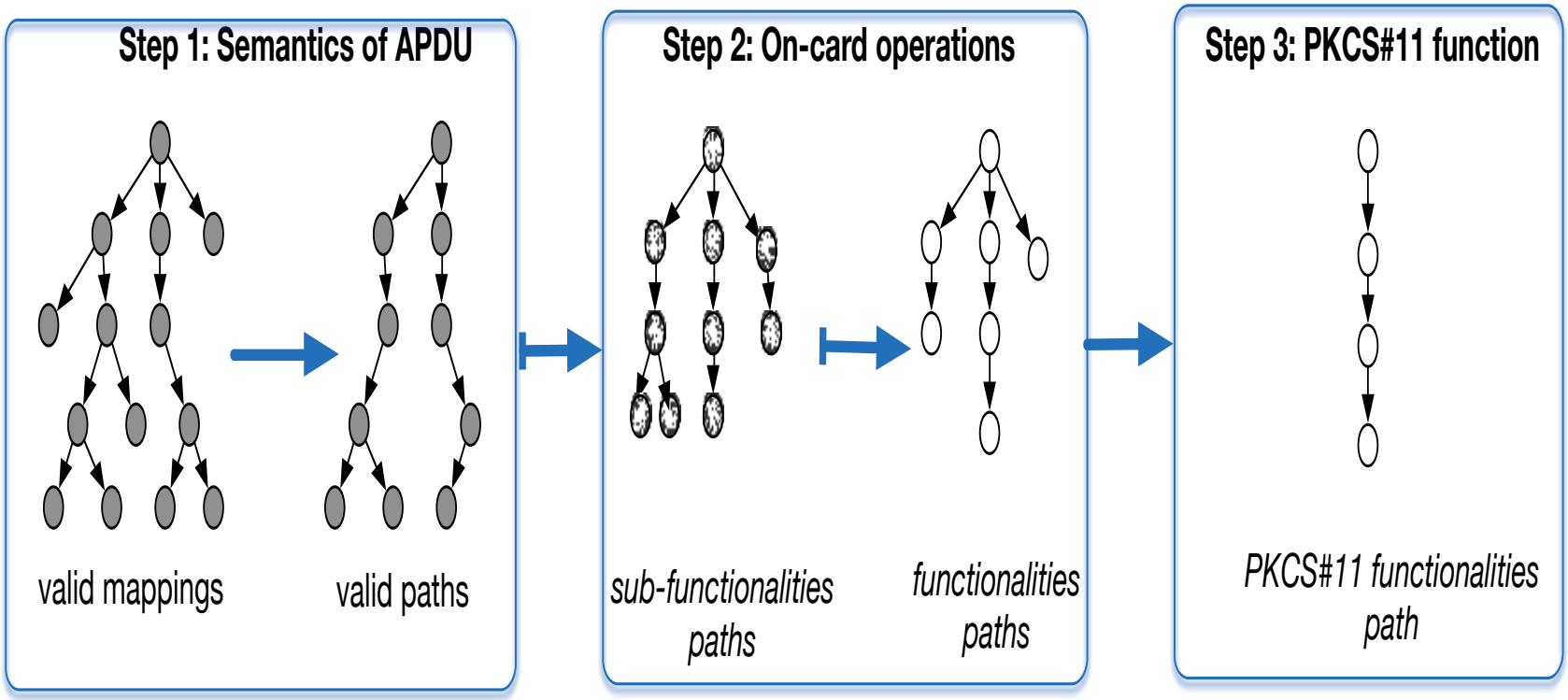


# APDU modelling

*PKCS#11 functions* are expressed as sets of `functionalities`
E.g., C_logIn :

- inputs/outputs specified by PKCS#11
- authentication as defined by ISO 7816
  - with key;
  - with PIN;
  - using internal data;
  - data encipherment
- additional operations
  - secondary authentication
  - data retrieval

# Reverse-engineering main idea



# Inferred model

*3* abstractions of the protocol $\mapsto$ *3* levels of attacks

# Inferred model

*3* abstractions of the protocol $\mapsto$ *3* levels of attacks

## Commands

semantics of the exchanged commands

- *identify sensitive data, inject commands, blind reply sessions*

# Inferred model

*3* abstractions of the protocol $\mapsto$ *3* levels of attacks

## Commands

semantics of the exchanged commands

- *identify sensitive data, inject commands, blind reply sessions*

## On-card operations

which/how on-card operations are executed

- *perform unauthorised operations*

# Inferred model

*3* abstractions of the protocol $\mapsto$ *3* levels of attacks

## Commands

semantics of the exchanged commands

- *identify sensitive data, inject commands, blind reply sessions*

## On-card operations

which/how on-card operations are executed

- *perform unauthorised operations*

## PKCS#11 interconnection

how a specific cryptographic function is executed at the APDU layer

- *PKCS#11 attacks*
- *bypass API restrictions*

# Inferred model: example

Sniffed trace:

```
>00a4080c0450154400
>9000
>800a0200ea
>Response
>00a4080c08501550724b025502
>9000
>80bb01b803840102
>9000
>80aa808602ffff
>Response
```

# Inferred model: example

```
*** trace translation
SELECT:00a4080c0450154400 ->
isa(50154400,df),select(file,50154400)

READ RECORD:800a0200ea ->
isa(02, offset),isa(Response, record),retrieve_data(ea, Response)

SELECT: 00a4080c08501550724b025502 ->
isa(501550724b025502, df),select(file,501550724b025502)

MANAGE SECURITY ENV: 80bb01b803840102 ->
set_security_env(840102)

PERFORM SECURITY OPERATION:80aa808602ffff ->
isa(80,tag),operation(ffff, Response)


*** operation steps
[read_data_sub(50154400, ea, Response)]
[security_env(840102),security_operation(ffff, Response)]


*** operations
data_retrieval(Response)
sign(ffff, Response)
```

# Experiments

Sniffed APDUs from 5 commercially available smart-cards;
9 PKCS#11 functions

- `C_logIn`
- `C_generateKey`
- `C_sign`
- `C_encrypt`
- `C_findObjects`
- `C_getAttributeValue`
- `C_setAttributeValue`
- `C_wrapKey`
- `C_unwrapKey`

# Experiments

evaluation on:

# Experiments

evaluation on:

- functional success

# Experiments

evaluation on:

- functional success
  - successfully inferred at least 1 model

# Experiments

evaluation on:

- functional success
  - successfully inferred at least 1 model
- quality of the results

# Experiments

evaluation on:

- functional success
  - successfully inferred at least 1 model
- quality of the results
  - apart from 3 cases; a *unique model* that matched *exactly*
  - 3 cases: *correct* on-card operations; 2 suggested models; 1 matched exactly

# Experiments

evaluation on:

- functional success
  - successfully inferred at least 1 model
- quality of the results
  - apart from 3 cases; a *unique model* that matched *exactly*
  - 3 cases: *correct* on-card operations; 2 suggested models; 1 matched exactly
- search-space restriction

# Experiments

evaluation on:

- functional success
  - successfully inferred at least 1 model
- quality of the results
  - apart from 3 cases; a *unique model* that matched *exactly*
  - 3 cases: *correct* on-card operations; 2 suggested models; 1 matched exactly
- search-space restriction
  - no explosion

# Search-space sample

| | Function | Total B.CC | R.CC | R.SFC | R.FC | R.Model |
|---|---|---|---|---|---|---|
| $\mathbf{Card}_2$ | `C_logIn` | 32000 | 12 | 4 | 2 | 2 |
| | `C_findObjects` | 400 | 3 | 1 | 1 | 1 |
| | `C_generateKey` | $540 \times 86^8$ | 512 | 69 | 8 | 1 |
| | `C_setAttributeValue` | 86 | 14 | 3 | 1 | 1 |
| | `C_encrypt` | 20 | 3 | 4 | 2 | 1 |
| $\mathbf{Card}_4$ | `C_logIn` | 7396 | 65 | 39 | 21 | 1 |
| | `C_findObjects` | 7396 | 6 | 1 | 1 | 1 |
| | `C_getAttributeValue` | 54700816 | 3 | 1 | 1 | 1 |
| | `C_sign` | 86 | 1 | 1 | 1 | 1 |
| $\mathbf{Card}_5$ | `C_logIn` | 1 | 1 | 1 | 1 | 1 |
| | `C_sign` | 12322 | 53 | 7 | 4 | 2 |
| | `C_setAttributeValue` | 1 | 1 | 1 | 1 | 1 |

*B.CC*: baseline algorithm command combinations.
*R.CC*: REPROVE command combinations.
*R.SFC*: REPROVE sub-functionality combinations,
*R.FC*: REPROVE functionality combinations.
*R.Model* is the final model(s) suggested by REPROVE.

# Results: Violations found

## c_logIn function

- No session handles
  - **all cards**
- No verification
  - **1 card**
- PIN sent in plaintext
  - **2 cards**

## c_wrapKey

- function executed library side $\mapsto$ sensitive key sent in plaintext
  - **1 card**

# Results: Violations found

## c_generateKey

- function executed library side ↦ sensitive key sent in plaintext
  - **2 cards**

## c_encrypt

- function executed library side ↦ sensitive key sent in plaintext
  - **1 cards**

- The location of the sensitive data and the related information (eg., attributes) was located for all cards.

# Conclusion

**REPROVE**: fully automated system for reverse-engineering APDUs and discovering interconnection with PKCS#11 functions

- it does not requires access to the card's code nor the API
- check if the card respects the standard
  - 2 tested cards did nothing!
- access PKCS#11 objects from the low-level - bypass API restrictions