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

Andriana Gkaniatsou\textsuperscript{1}, Fiona McNeill\textsuperscript{2}, Alan Bundy\textsuperscript{1}, Graham Steel\textsuperscript{3}, Riccardo Focardi\textsuperscript{4}, Claudio Bozzato\textsuperscript{4}

\textsuperscript{1}University of Edinburgh \textsuperscript{2}Heriot-Watt \textsuperscript{3}Cryptosense \textsuperscript{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

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

APDU Command
AF 03 16 78

APDU Response
90 00

Host

Smart Card Reader

APDU Layer
Physical Layer
(serial, usb, bluetooth, ...)

Andriana Gkaniatsou
Getting to know your card
The REPROVE system

REPROVE reverse-engineering system: no API access - no card access - implementation independent

- APDU semantics
- On-card operations
- PKCS#11 function translation

PKCS#11
ISO 7816
generic
assumptions
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

Analysis Challenge

How can we infer the semantics of the proprietary command?

e.g., 21 \rightarrow a4?
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

Analysis Challenge
How can we infer the semantics of the proprietary command? e.g., 21 ↦ a4?
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  >80 21 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

**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**

- **operation** decomposed into **steps** implemented as **APDU layer commands** characterised by **data exchange and role properties**

- **sub-functionality**
  - **sub-functionality_1**
    - command_a
    - command_b
  - **sub-functionality_2**
    - command_x
    - command_y
    - command_z
  - **sub-functionality_3**
    - command_a
    - command_x

- Additional roles:
  - YY, core
  - NY, core
  - YY, core
  - NY, additional
  - YN, additional
  - YY, additional
  - NN, dummy
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

Step 1: Semantics of APDU
- Valid mappings
- Valid paths

Step 2: On-card operations
- Sub-functionalities paths
- Functionalities paths

Step 3: PKCS#11 function
- PKCS#11 functionalities path

Andriana Gkaniatsou
Getting to know your card
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

- \textit{identify sensitive data, inject commands, blind reply sessions}
Inferred model

3 abstractions of the protocol \(\leftrightarrow\) 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:
>00a4080c045015400
>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

<table>
<thead>
<tr>
<th>Function</th>
<th>Total B.CC</th>
<th>R.CC</th>
<th>R.SFC</th>
<th>R.FC</th>
<th>R.Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Card₂ C_logIn</td>
<td>32000</td>
<td>12</td>
<td>4</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>C_findObjects</td>
<td>400</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>C_generateKey</td>
<td>540x86</td>
<td>512</td>
<td>69</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>C_setAttributeValue</td>
<td>86</td>
<td>14</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>C_encrypt</td>
<td>20</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Card₄ C_logIn</td>
<td>7396</td>
<td>65</td>
<td>39</td>
<td>21</td>
<td>1</td>
</tr>
<tr>
<td>C_findObjects</td>
<td>7396</td>
<td>6</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>C_getAttributeValue</td>
<td>54700816</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>C_sign</td>
<td>86</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Card₅ C_logIn</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>C_sign</td>
<td>12322</td>
<td>53</td>
<td>7</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>C_setAttributeValue</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

*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 $\rightarrow$ sensitive key sent in plaintext
  - 1 card
Results: Violations found

**c_generateKey**
- function executed library side $\mapsto$ sensitive key sent in plaintext
  - 2 cards

**c_encrypt**
- function executed library side $\mapsto$ 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