Vulnerability Analysis on Smart Cards using Fault Tree

Guillaume Bouffard Bhagyalekshmy N Thampi
Jean-Louis Lanet

Smart Secure Devices (SSD) Team – XLIM/University of Limoges
guillaume.bouffard@xlim.fr
http://secinfo.msi.unilim.fr

SAFECOMP 2013

INOSSEM

Outline

Université de Limoges

The Smart Card

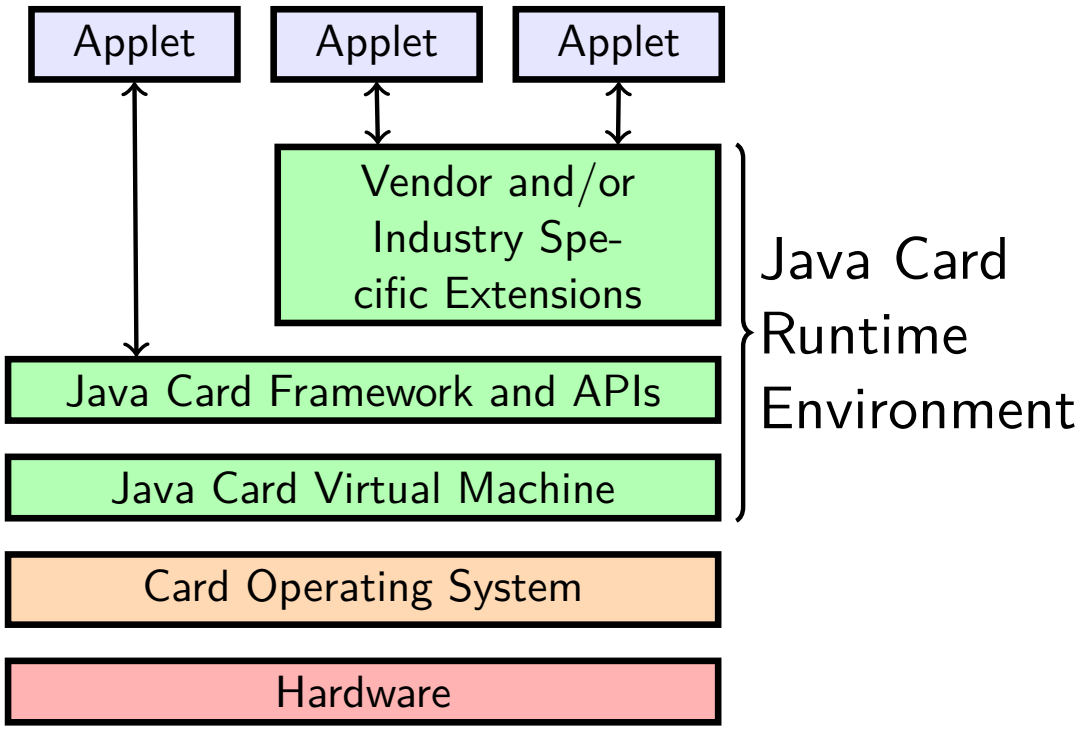
Widely used device

- Credit Card;
- (U)SIM Card;
- Health Card (french Vitale card);
- Pay TV;
- ...

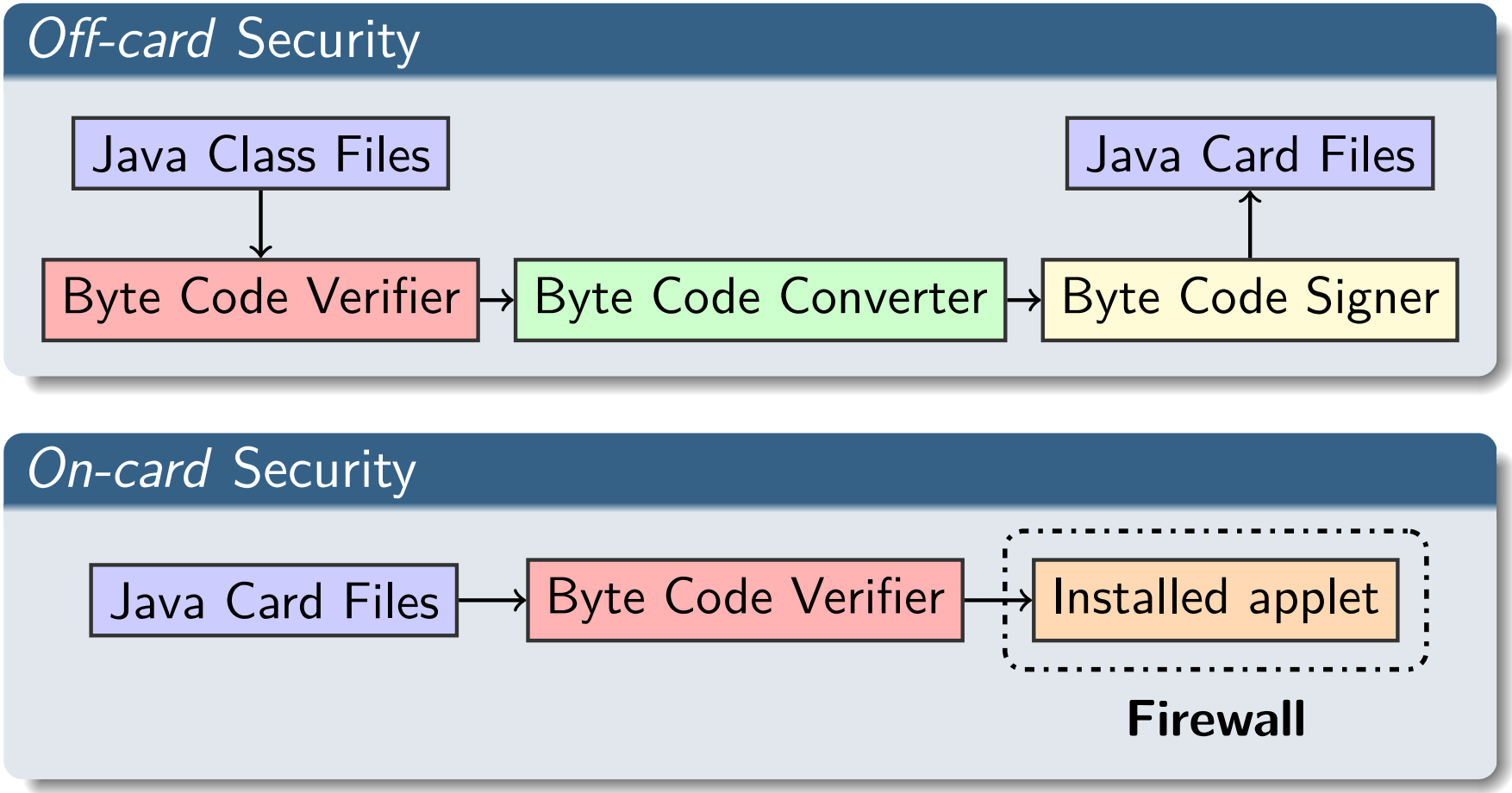
This device contains sensitive data

Java Card based Smart Card

- Created by Schlumberger in 1996.
- **Specified** by Oracle
- Provide a **friendly** environment to develop **secure** Java-applications.



Java Card Security Model



Java Card Attacks

Logical attacks

- Execution of malicious Java Card byte codes

Physical attacks

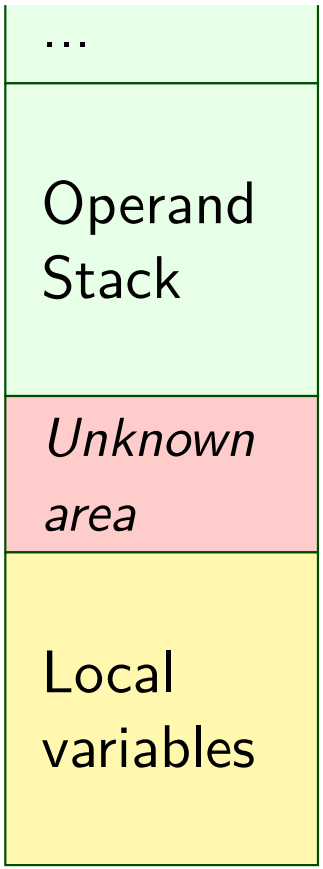
- Side Channel attacks (timing attacks, power analysis attack, ...);
- Fault attacks (electromagnetic injection, laser beam injection, ...)

Combined attacks

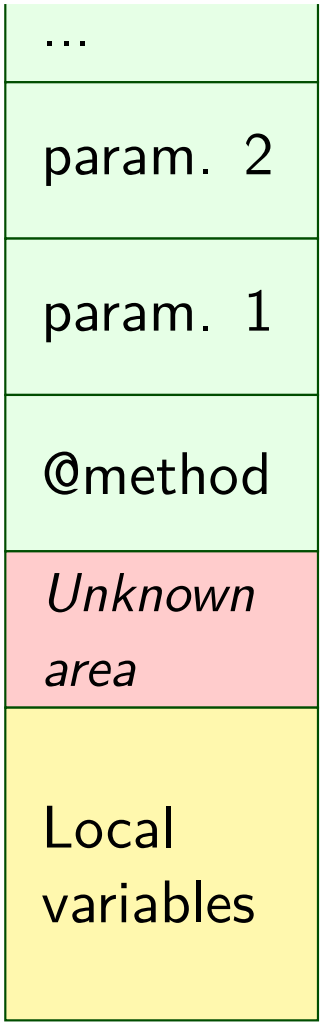
Mix of physical and logical attacks.

Université de Limoges

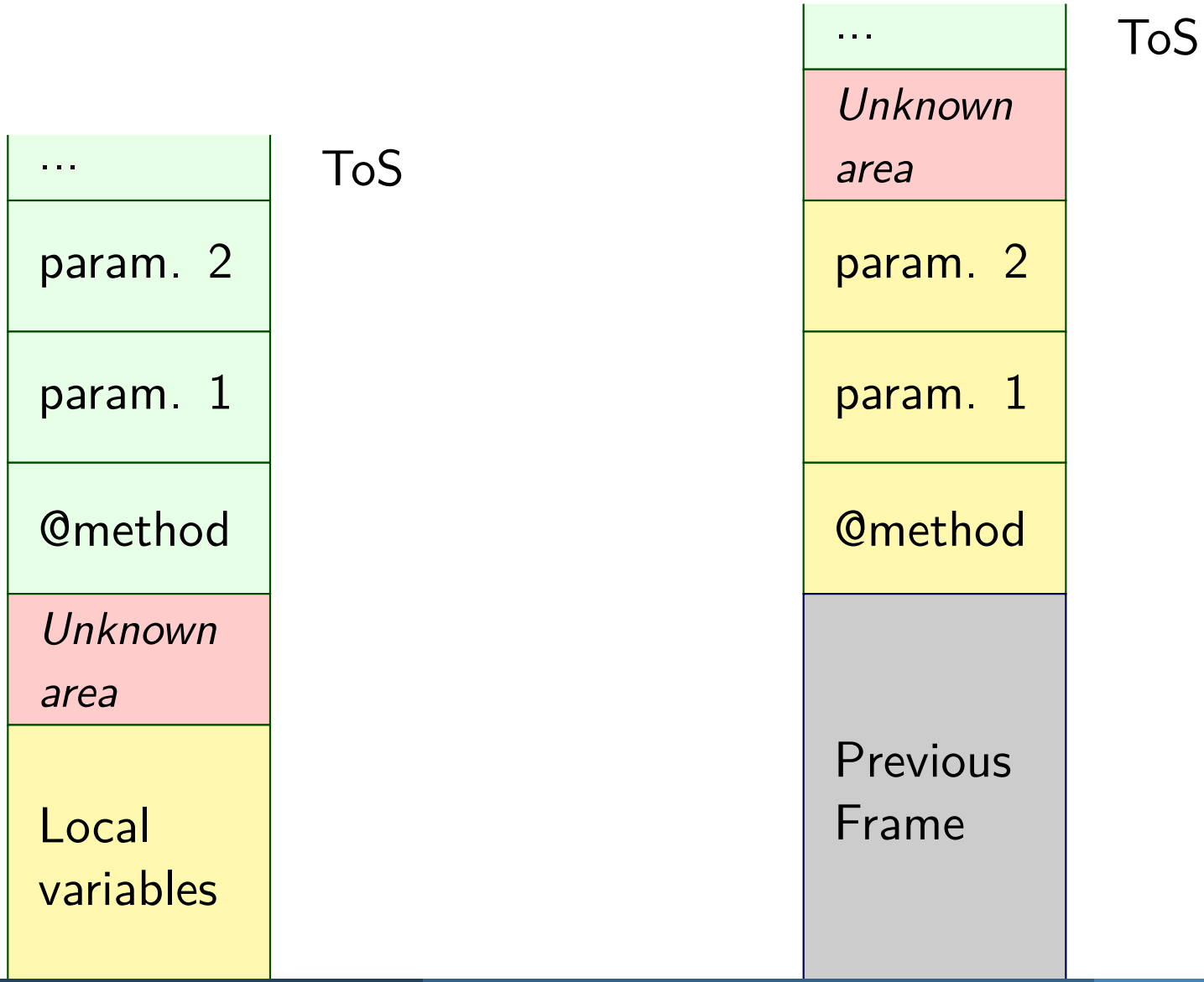
Example: EMAN 2 attack



Example: EMAN 2 attack



Example: EMAN 2 attack

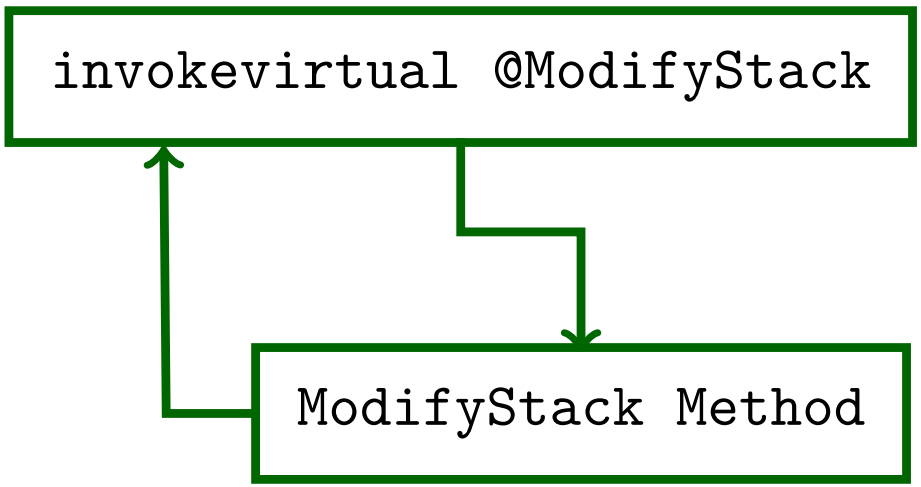


Example: EMAN 2 attack

```
public void ModifyStack
      (byte[] apduBuffer,
       APDU apdu,
       short a) {
   short i=(short)0xCAFE;
   short j=(short)
       maliciousFunction();
   i = j ;
}
```

Example: EMAN 2 attack

```
public void ModifyStack
      (byte[] apduBuffer,
       APDU apdu,
       short a) {
   short i=(short)0xCAFE;
   short j=(short)
       maliciousFunction();
   i = j ;
}
```

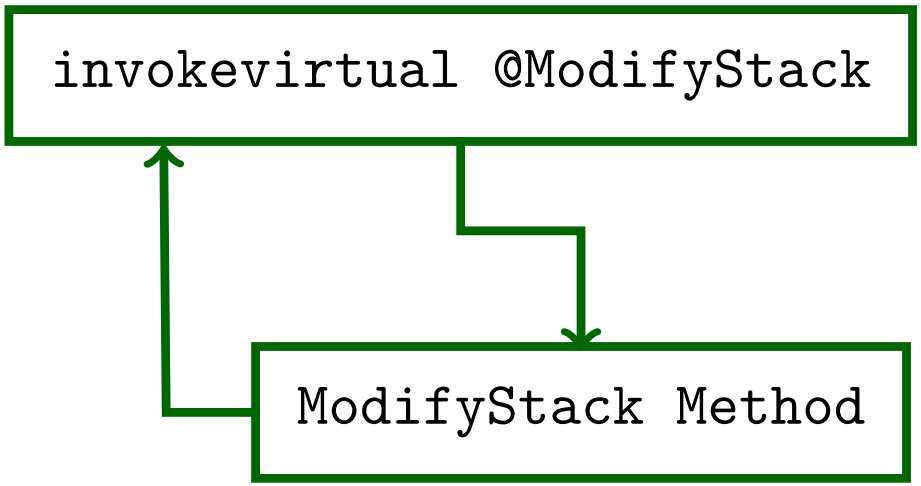


Any unchecked byte code

Université de Limoges

Example: EMAN 2 attack

```
public void ModifyStack
    (byte[] apduBuffer,
     APDU apdu, short a)
{ 02 // flags:0 max_stack:2
  42 // nargs:4 max_locals:2
  11 CAFE   sspush        0xCAFE
  29 04     sstore          4
  18        aload_0
  7B 00     getstatic_a     0
  8B 01     invokevirtual   1
  29 05     sstore          5
  16 05     sload           5
  29 04     sstore          4
  7A        return            }
```



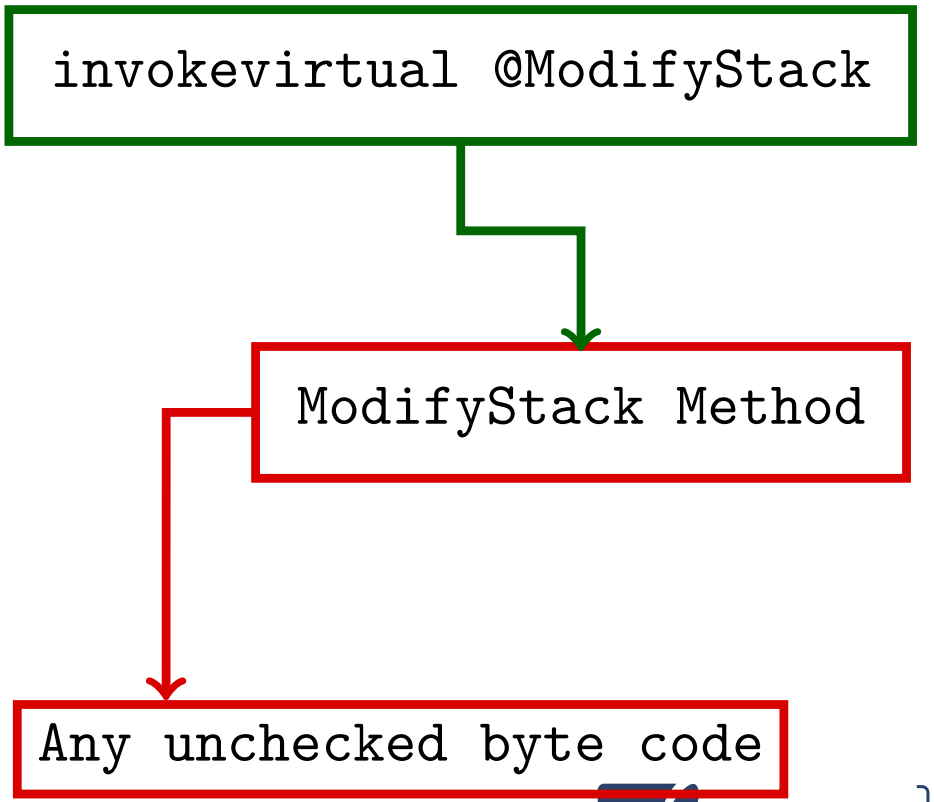
Any unchecked byte code

Université de Limoges

Example: EMAN 2 attack

```
public void ModifyStack
    (byte[] apduBuffer,
     APDU apdu, short a)
{ 02 // flags:0 max_stack:2
  42 // nargs:4 max_locals:2
  11 CAFE   sspush        0xCAFE
  29 04     sstore          4
  18        aload_0
  7B 00     getstatic_a     0
  8B 01     invokevirtual   1
  29 05     sstore          5
  16 05     sload           5
  29 07     sstore          7
  7A        return            }
```

The Return Address of the current function is changed!



Université de Limoges

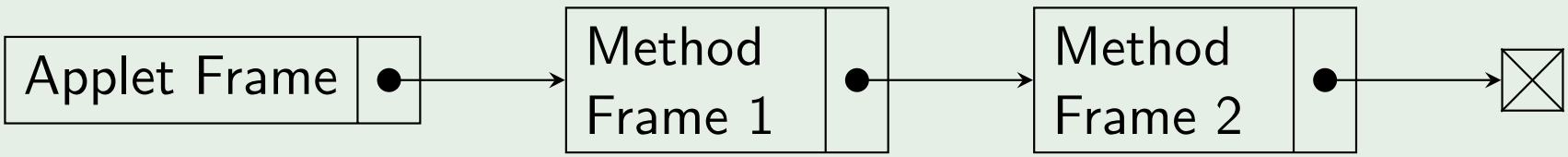
Example: EMAN 2 attack – Countermeasure

Security Requirements

- Embed a **B**yte **C**ode **V**erifier (BCV);
- Check the number of locals;
- Check the frame integrity;

Proposed Countermeasure: the *linked-frame*

- The memory area is non-contiguous
- The top of stack should be copied



Université de Limoges

Problematic

Inductive Approach

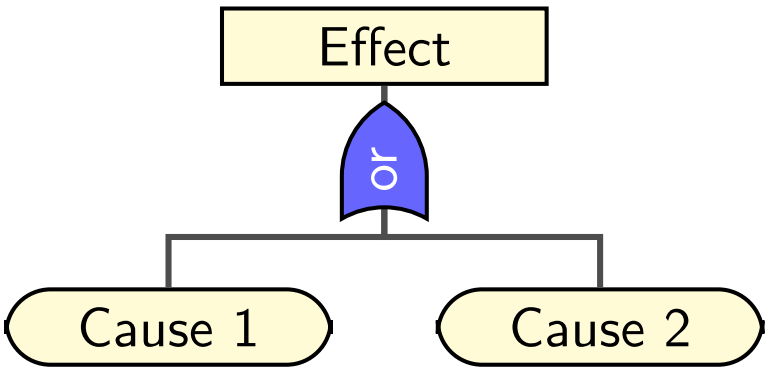
- 1 attack = 1 countermeasure
- bottom-up solution

Our Requirements

- A top-down analytic solution;
- Definition of each undesirable events;

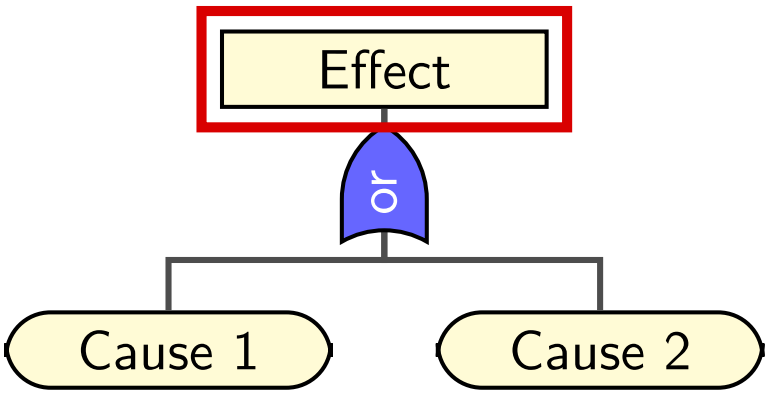
Université de Limoges

The Fault Tree Analysis (FTA)



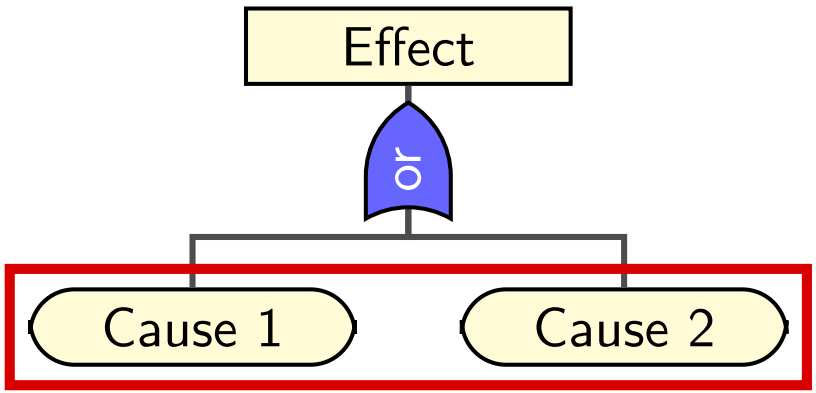
- Undesirable event;
- Initial causes;
- Gate Connector.

The Fault Tree Analysis (FTA)



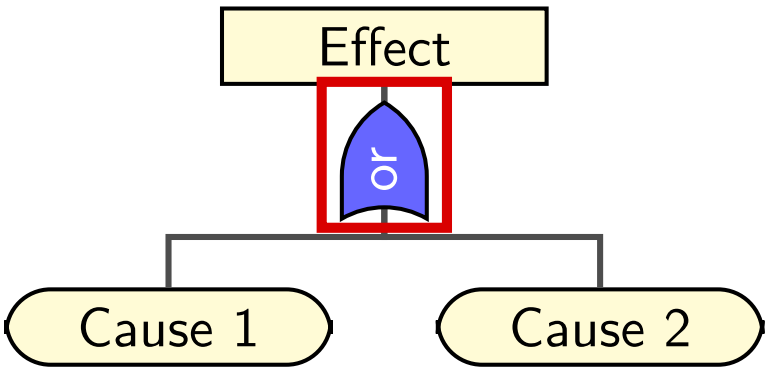
- **Undesirable event;**
- Initial causes;
- Gate Connector.

The Fault Tree Analysis (FTA)



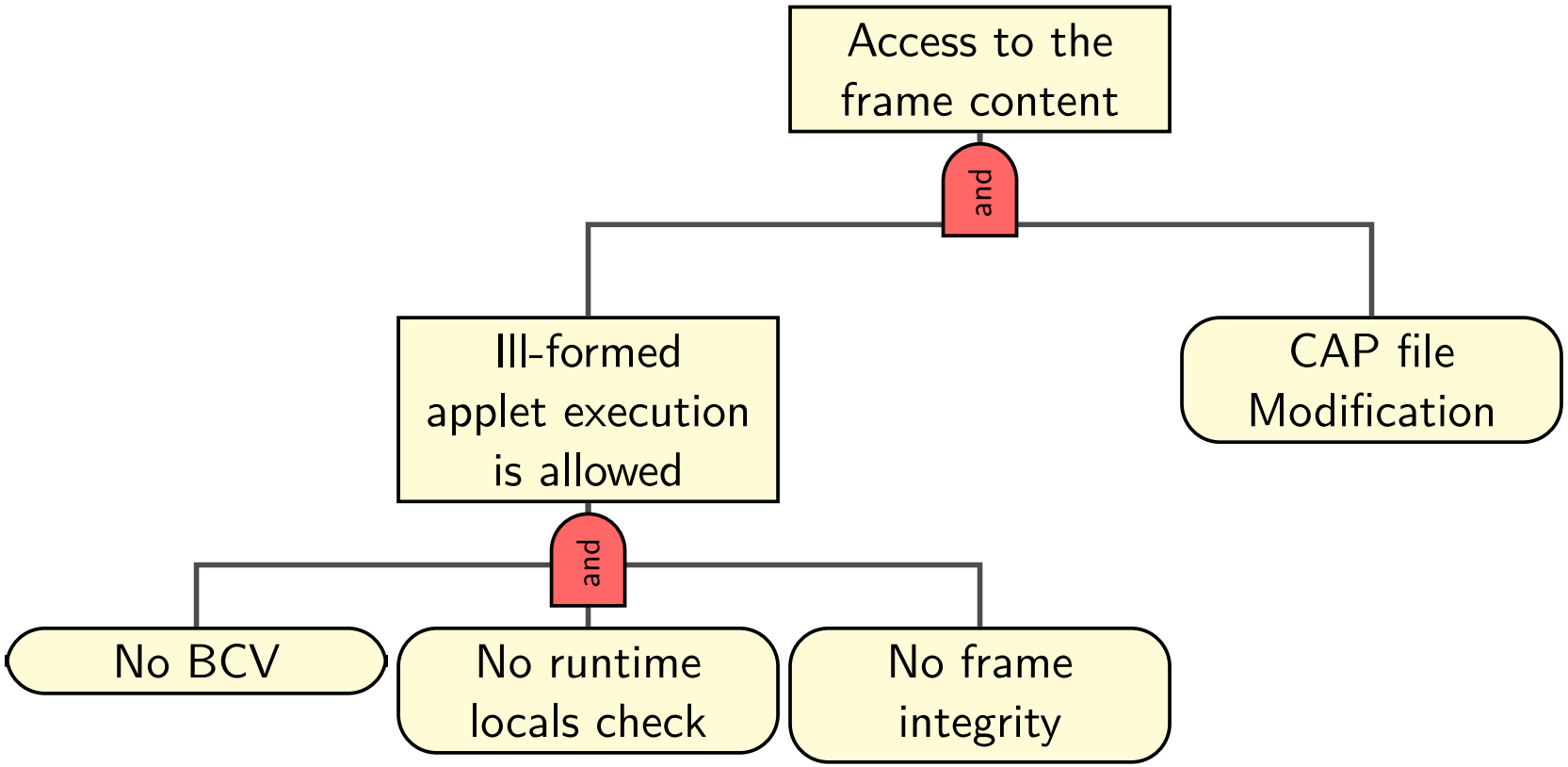
- Undesirable event;
- **Initial causes;**
- Gate Connector.

The Fault Tree Analysis (FTA)



- Undesirable event;
- Initial causes;
- **Gate Connector.**

FTA for EMAN 2 attack



Smart Card's Assets

Undesirable events can affect:

- Code integrity;
- Data integrity;
- Code confidentiality;
- Data confidentiality;

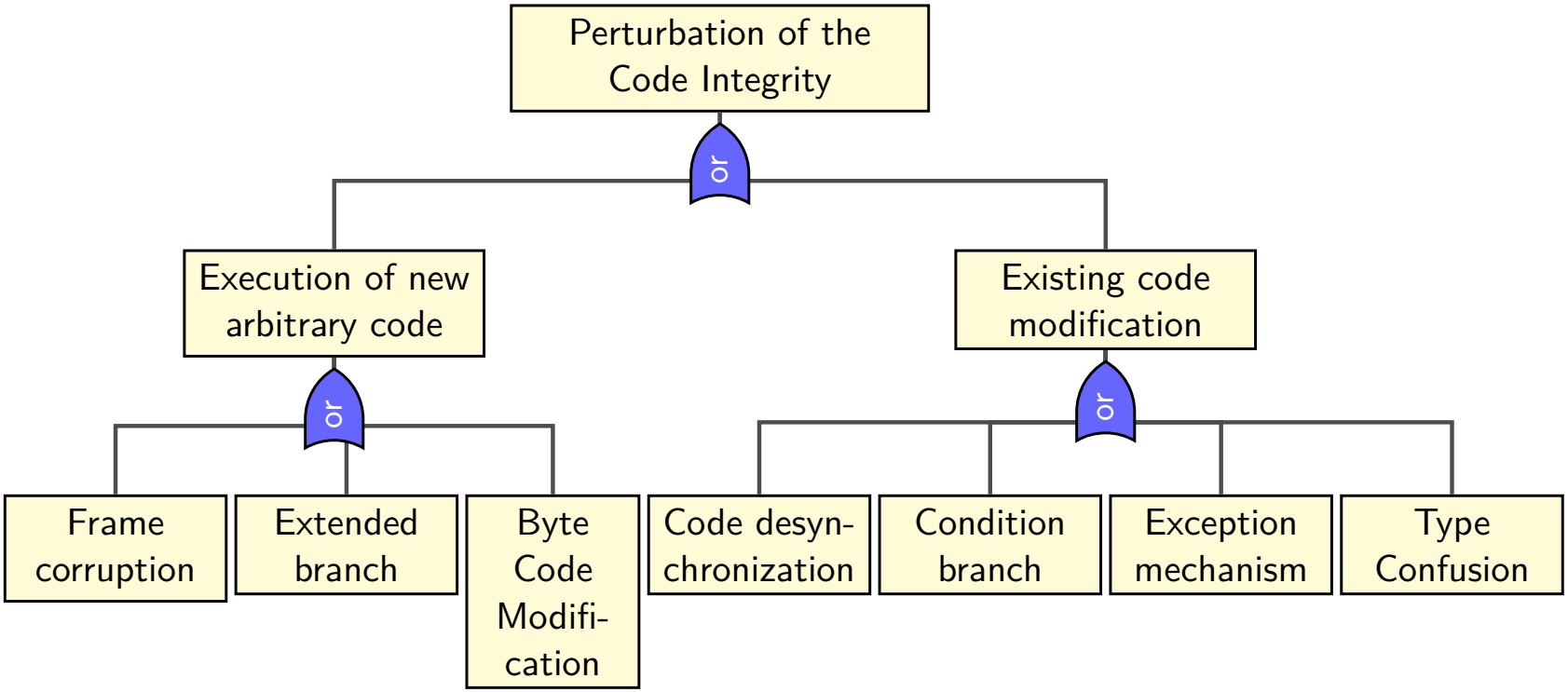
Smart Card's Assets

Undesirable events can affect:

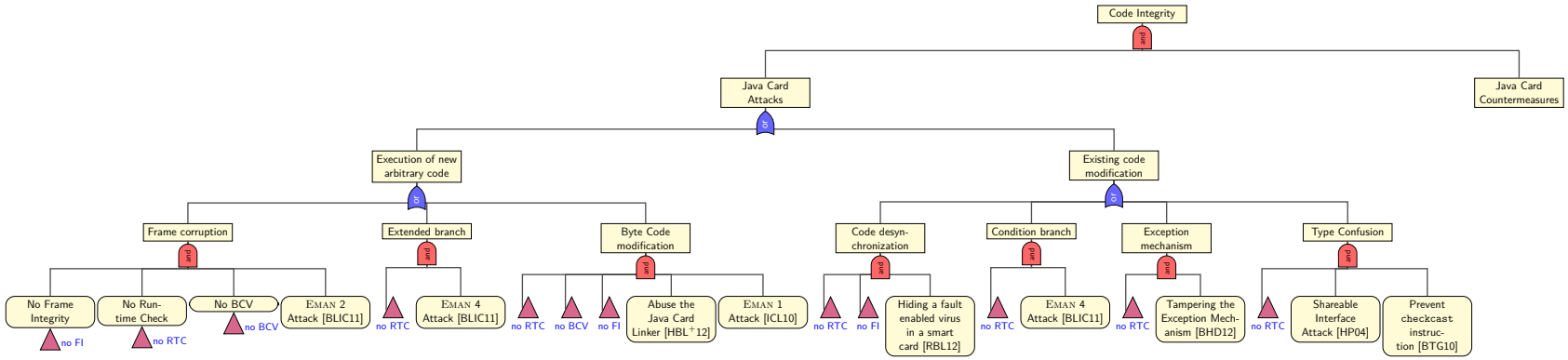
- **Code integrity**;
- Data integrity;
- Code confidentiality;
- Data confidentiality;

An attack offers the execution of a malicious byte code.

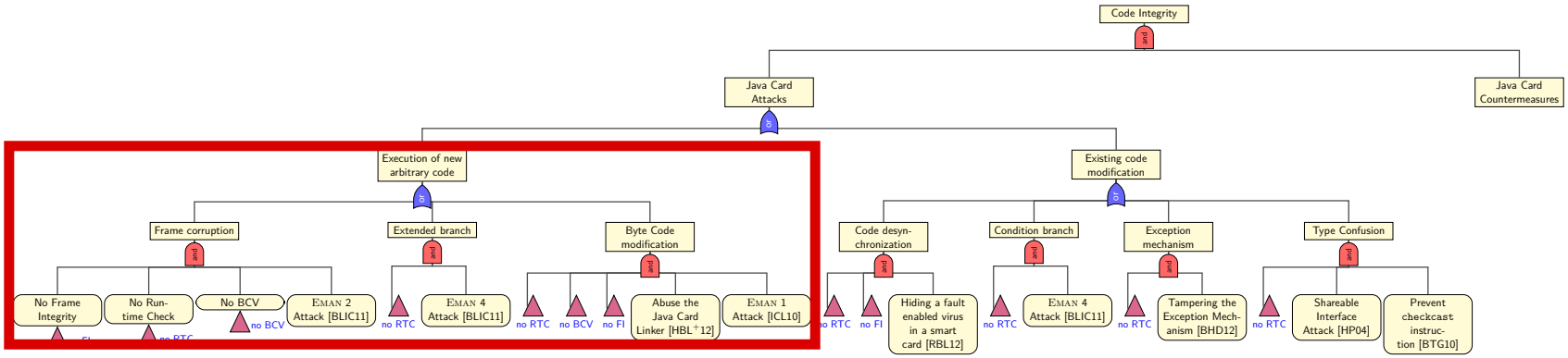
How to break the Java Card Code Integrity?



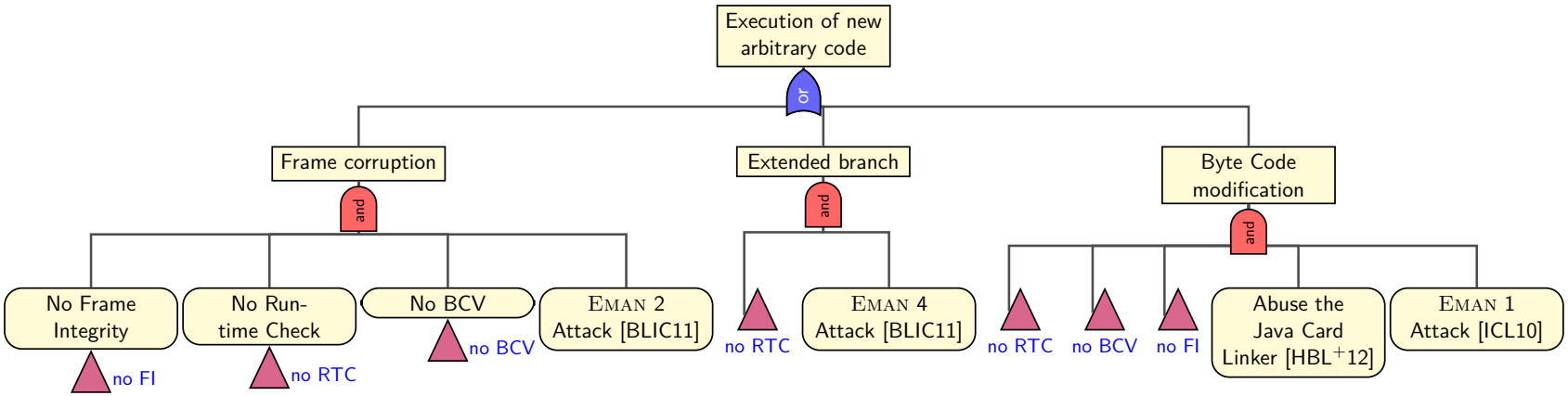
Code Integrity Tree



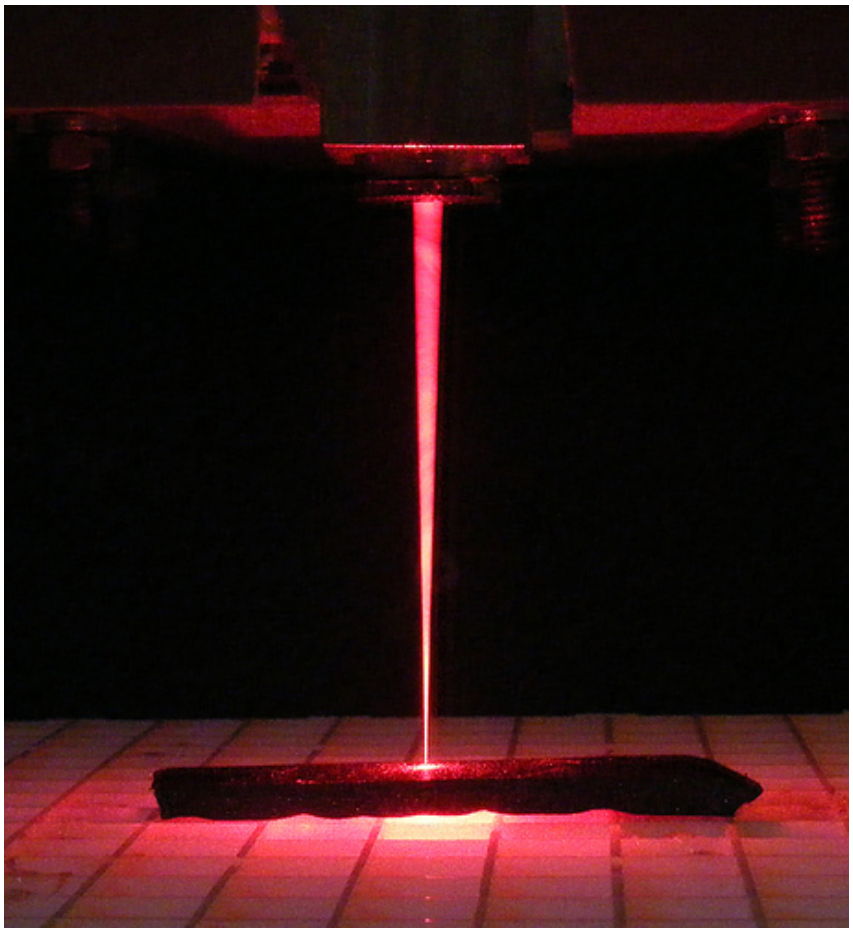
Code Integrity Tree



Code Integrity Tree



New Undesirable Events detected



Laser beam injection effects

- Precise byte errors into the memory;
- Perturb the runtime execution.

Laser beam injection issues

- Modification of the returned value;
- Bypass a method calls;
- Modify the data's address.

Université de Limoges

Principle

Countermeasures should be implemented in the:

- application level;
- Java Card Virtual Machine level.

Principle

Countermeasures should be implemented in the:

- **application level**;
- Java Card Virtual Machine level.

Advantages

- Implementation of several checks;
- The developer knows the assets to protect.

Drawbacks

- Code redundancy;
- Increase the program's size.

Université de Limoges

Principle

Countermeasures should be implemented in the:

- application level;
- **Java Card Virtual Machine level**.

Advantages

- Low level system countermeasures;
- Stored in ROM module.

Université de Limoges

INOSSEM API

"The aim [of the INOSSEM project] is to provide security interoperability between smart card manufacturers."

INOSSEM

The INOSSEM API defines functions to protect:

- Code Integrity;
- Frame Integrity;
- Data Integrity.

Université de Limoges

Frame Integrity Protection

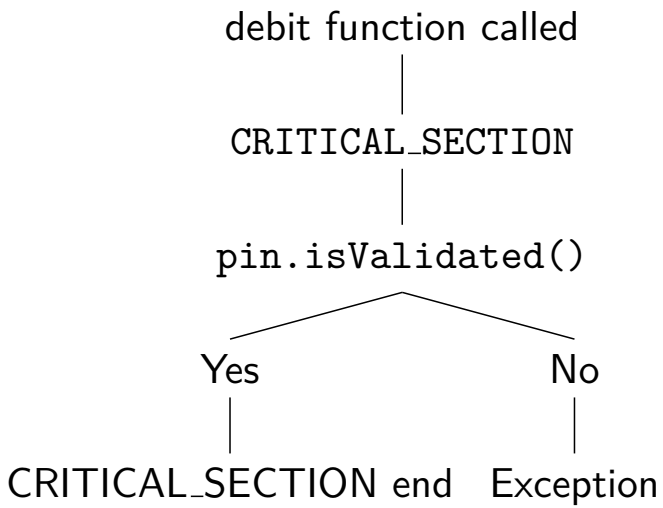
```
// initialize the signature object with the first
    parameter
paramChkInstance.init(firstParam);
// update signature buffer
paramChkInstance.update(secondParam);
// sign parameters
short paramChk = paramChkInstance.doFinal(userPIN);
// invoke the sensitive method
sensitiveMeth(firstParam, secondParam,
              userPIN,    paramChk);
```

Université de Limoges

JPC Protection

The **J**ava **P**rogram **C**ounter (JPC) can be **corrupted** by a **laser beam injection**.

```
private void debit(APDU apdu) {
// transition to a new state
this.setState(CRITICAL_SECTION);
if (!pin.isValidated()) {
 this.endStateMachine
  (PIN_VERIFICATION_REQUIRED_STATE);
 ISOException.throwIt
  (SW_PIN_VERIFICATION_REQUIRED);}
```



Université de Limoges

Conclusion

- FTA can be used for safety and smart card vulnerabilities analysis;
- New undesirable events were found;
- Detected undesirable events are prevented by the INOSSEM API;
- Next step: quantify the attacker's power

Université de Limoges

Thank you for your attention!
Do you have any questions?

guillaume.bouffard@xlim.fr
http://secinfo.msi.unilim.fr

Université de Limoges

References I

Guillaume Barbu, Philippe Hoogvorst, and Guillaume Duc.
Tampering with Java Card Exceptions - The Exception Proves the Rule.
In Pierangela Samarati, Wenjing Lou, and Jianying Zhou, editors, *SECRYPT*, pages 55–63. SciTePress, 2012.

Guillaume Bouffard, Jean-Louis Lanet, and Julien Iguchy-Cartigny.
Combined Software and Hardware Attacks on the Java Card Control Flow.
In *Smart Card Research and Advanced Applications*, volume 7079 of *Lecture Notes in Computer Science*, pages 283–296, Berlin / Heidelberg, 2011. Springer.

Université de Limoges

References II

Guillaume Barbu, Hugues Thiebeauld, and Vincent Guerin.
Attacks on Java Card 3.0 Combining Fault and Logical Attacks.
In Dieter Gollmann, Jean-Louis Lanet, and Julien Iguchi-Cartigny, editors, *CARDIS*, volume 6035 of *Lecture Notes in Computer Science*, pages 148–163. Springer, 2010.

Samiya Hamadouche, Guillaume Bouffard, Jean-Louis Lanet, Bruno Dorsemaine, Bastien Nouhant, Alexandre Magloire, and Arnaud Reygnaud.
Subverting Byte Code Linker service to characterize Java Card API.
In *Seventh Conference on Network and Information Systems Security (SAR-SSI)*, pages 75–81, May 22rd to 25th 2012.

Université de Limoges

References III

Engelbert Hubbers and Erik Poll.
Transactions and non-atomic API methods in Java Card: specification ambiguity and strange implementation behaviours.
Technical Report NIII-R0438, Radboud University Nijmegen, 2004.

Julien Iguchi-Cartigny and Jean-Louis Lanet.
Developing a trojan applets in a smart card.
Journal in Computer Virology, 6(4):343–351, 2010.

Université de Limoges

References IV

Tiana Razafindralambo, Guillaume Bouffard, and Jean-Louis Lanet.
A Friendly Framework for Hidding fault enabled virus for Java Based Smartcard.
In Nora Cuppens-Boulahia, Frédéric Cuppens, and Joaquín García-Alfaro, editors, *DBSec*, volume 7371 of *Lecture Notes in Computer Science*, pages 122–128. Springer, 2012.

Université de Limoges