JAVA CARD SECURITY

Introduction and attacks

October 21, 2014

Sergi Casanova scasanova@bitwise.cat
Pol Matamoros pol@bitwise.cat

bitwise.cat

OVERVIEW

1. Introduction Java Cards
2. Java Card Threats
3. Practical attacks

BITWISE

- → Bitwise is focused on embedded security.
- → The team comprises of developers and security analysts.
- → Embedded security projects: Common Criteria, banking payment, development.
- → Our areas of experise are:

Javacard Native cards Global Platform EMV
Mobile Payment Cryptography Side-channel
Fault injection Software attacks Malicious code
Common Criteria

OVERVIEW

- → Software technology that allows Java-based applications (AKA applets) to run securely on smart cards and similar small memory footprint devices.
- → Java Card is the tiniest of Java platforms targeted for embedded devices.
- → Gives the user the ability to program the devices and make them application specific.

- → In 1970 the smart card "concept" was patented.
- → German concept accomplished by French companies.
- → Patents property of Gemalto.
- → Closed hardware and software specifications (under personal responsibility).
- → Dedicated cryptographic HW

- → Platform specific code.
- → Machine code burned in ROM.
- → Direct access to HW functionality
- → Multi-application behavior relies on each developer

- New VM based technologies: Java Card, MultOS and Basic Card.
- Increase development complexity.
- IC functionalities accessed through an API.
- Multi-application platforms:
  - different / competing entities in the same card.
  - post-issuance card management
- Multi-platform applications: same weakness in multiple products and countries.
- Process efficiency reduction due to interpreted language.
- Compatible with existing Smart Card standards.

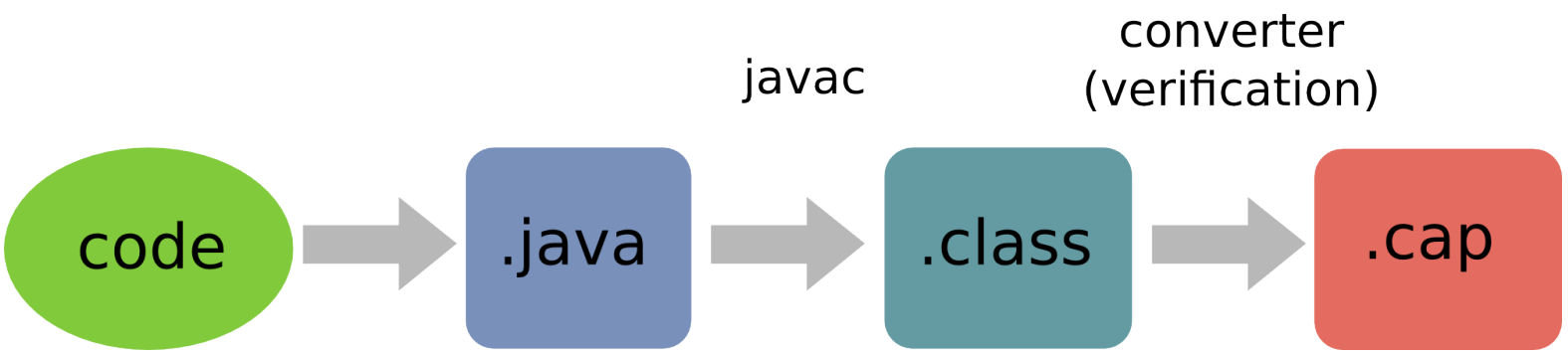
Feature	Value
ROM	32KB-400KB
RAM	around 32KB
EEPROM	around 100KB
FLASH	usually 1MB
CPU	around 50MHz

INTRODUCTION
USE CASES

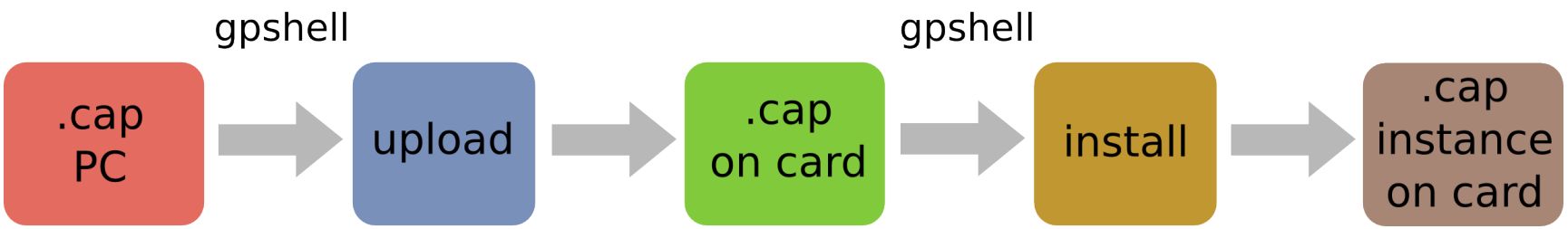
- → Introduced in 1996 motivated by the popularity of Java.
- → Object oriented programming language.
- → Java Card is a subset of Java.
- → Provides additional features that Java don't.
- → Implemented with GlobalPlatform for card management.

Feature	Java	Java Card
Code verification	Runtime	Off-card & on-card
Object persistency	No	Yes
Dynamic class loading	Yes	No
Multithreathing	Yes	No
Typesafe enumerations	Yes	No
Assertions	Yes	No
Atomic operations	Yes	Enhanced
Applet isolation	Yes	Yes, includes firewall
Garbage collector	Yes	No

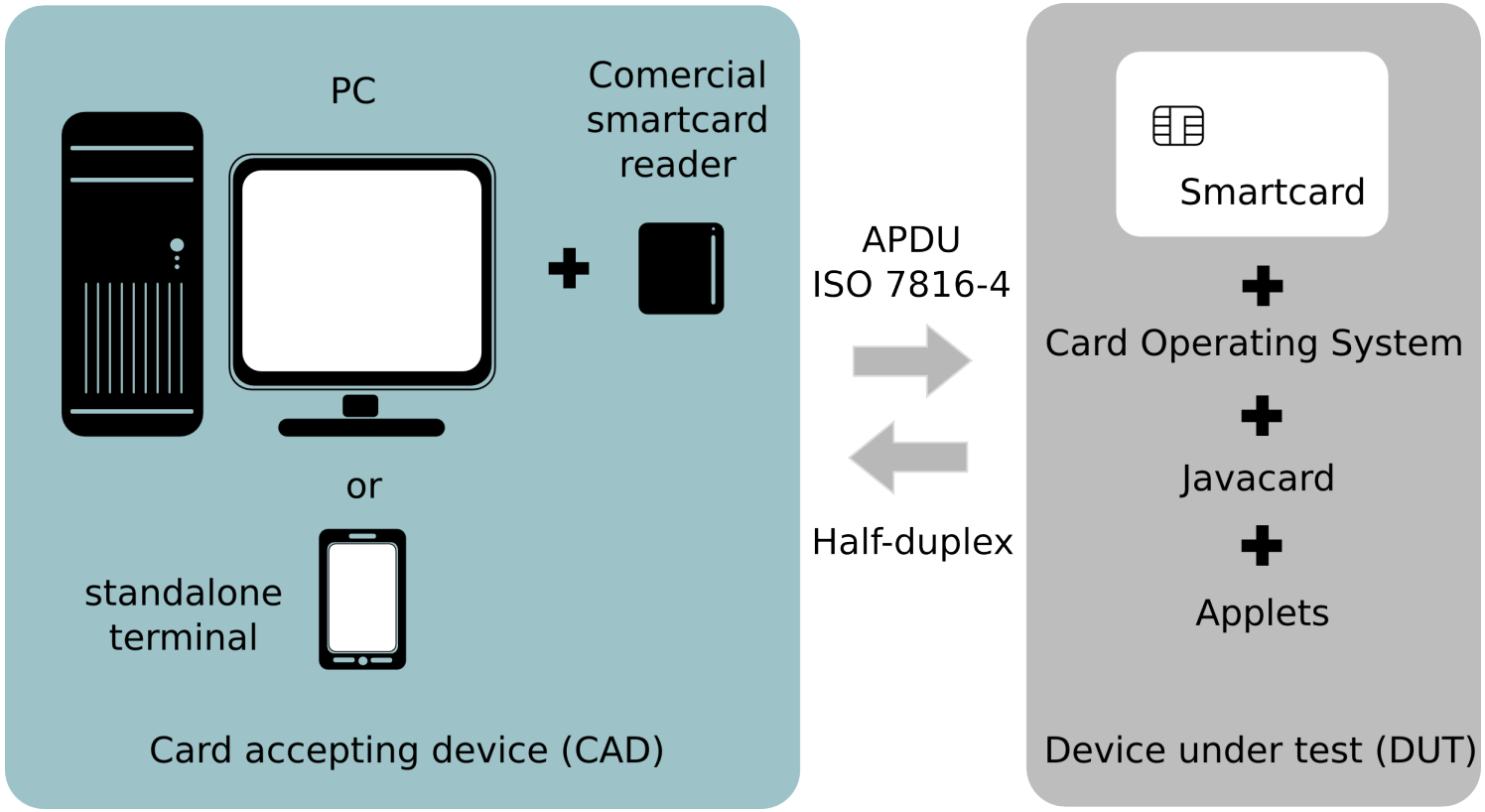
BASIC CONCEPTS
JAVACARD EXECUTABLE GENERATION



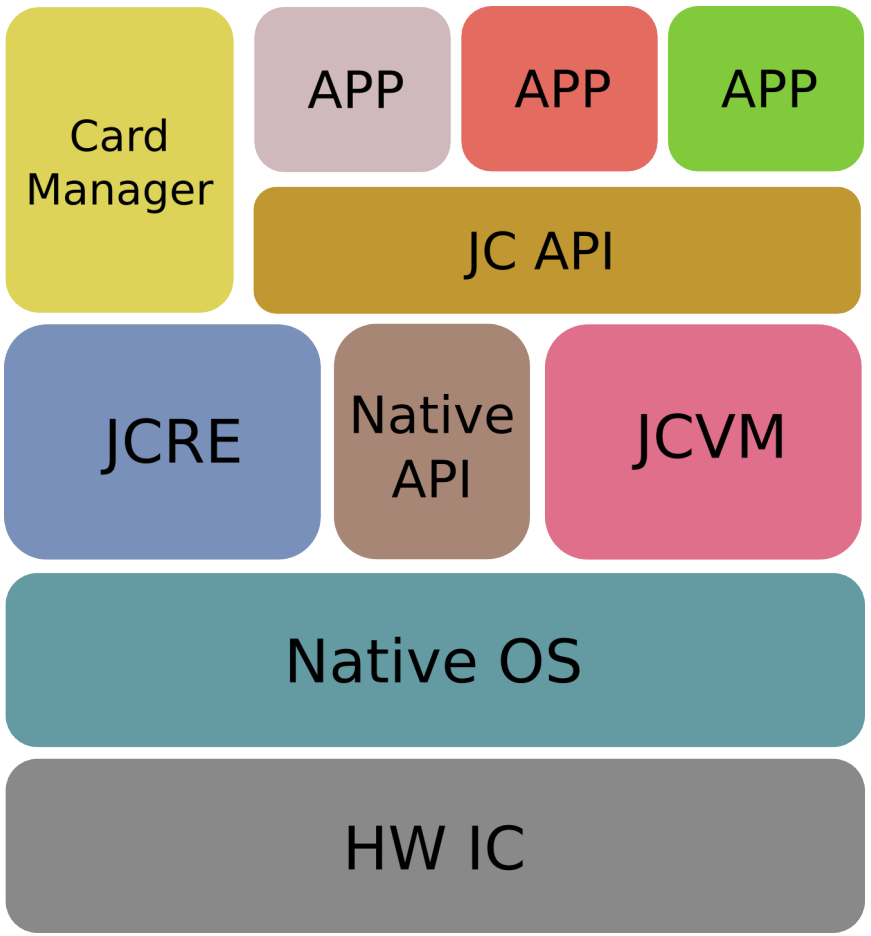
BASIC CONCEPTS
APPLET LOADING AND INSTANTATION



BASIC CONCEPTS
JAVACARD APPLICATION USAGE SCENARIO



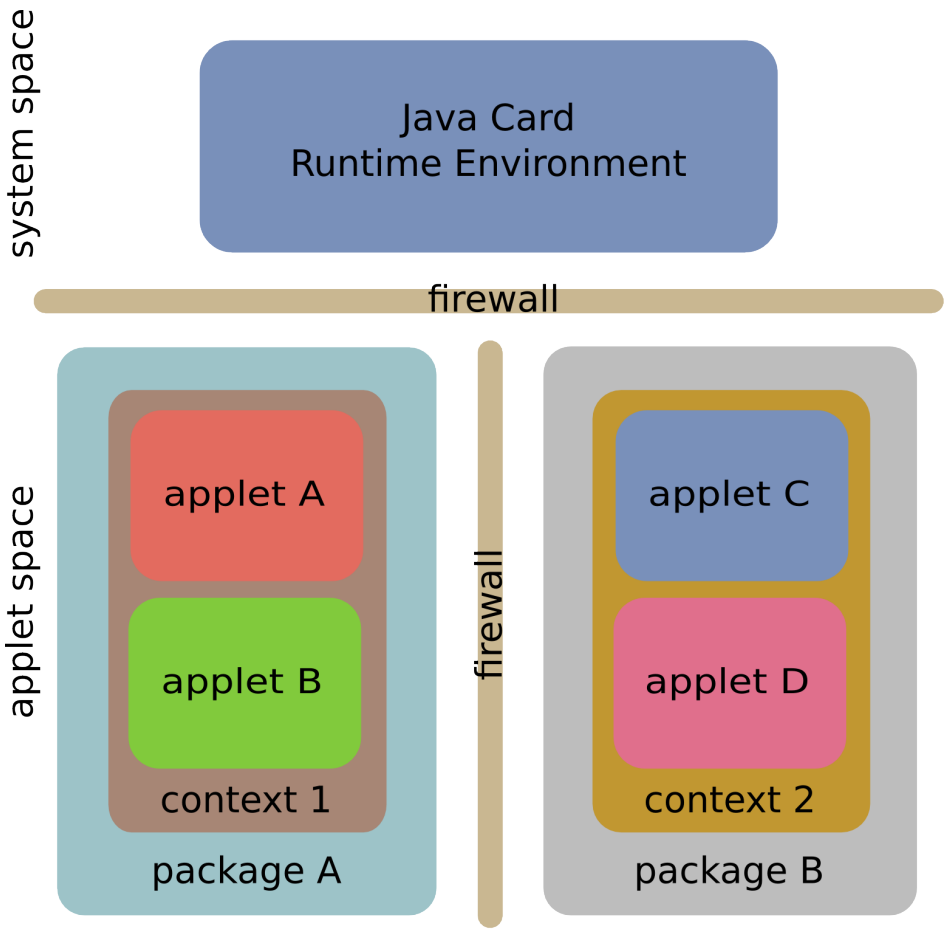
BASIC CONCEPTS
JAVA CARD ARCHITECTURE



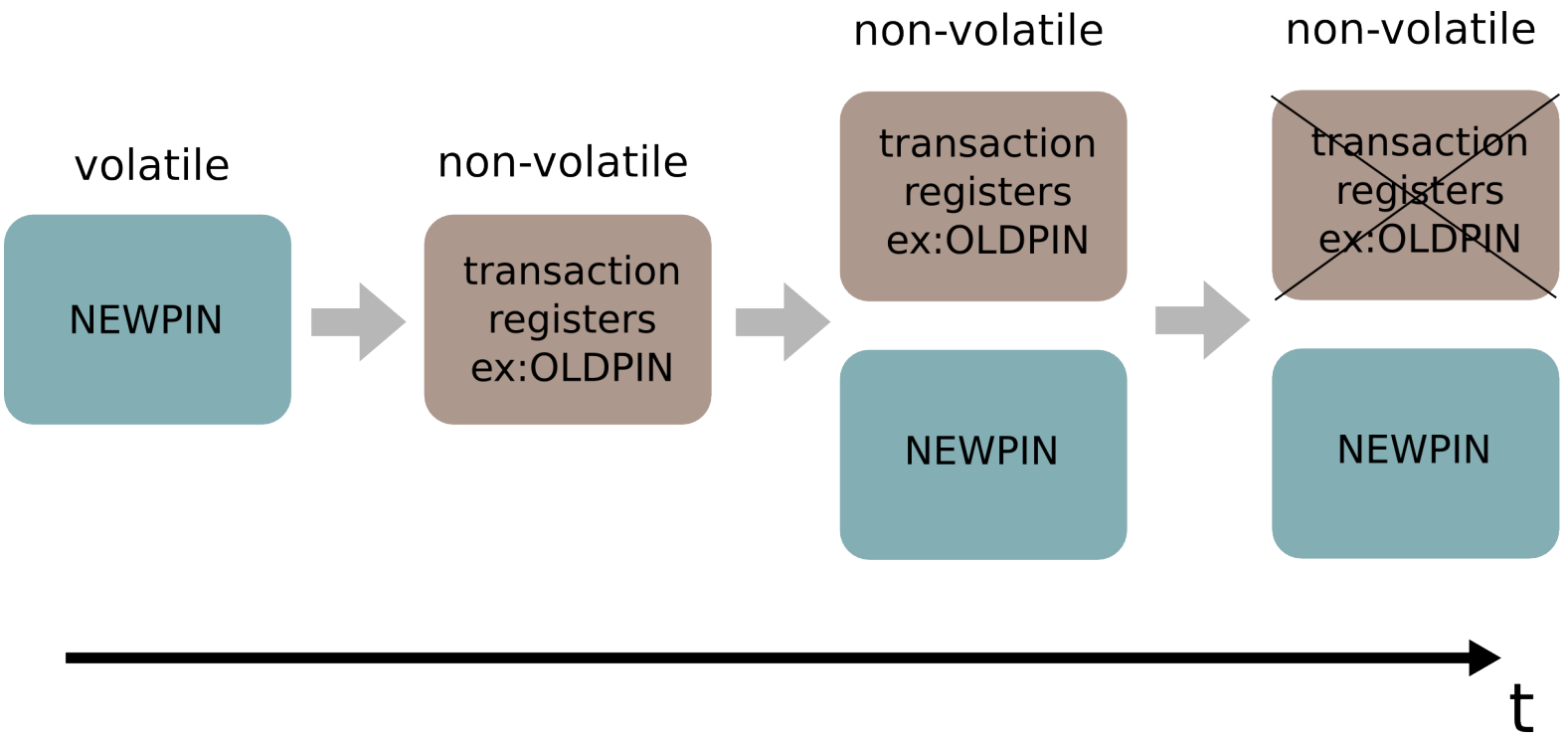
- JCRE provides a set of system classes and libraries.
- Provides an execution environment for the applets.
- Provides access to the native system features.
- Among others JCRE is in charge of:
  - JC applet lifetime.
  - Logical channels and applet selection.
  - Persistence management.
  - Execution context managing.
  - Applet isolation (firewall)

- → Responsible of the applet execution.
- → JCVM interpretes bytecode stored in applets sequentially as a generic OS.
- → JCVM specification define 186 out of the 255 possible opcodes.
- → JCVM keeps the two last bytecodes to:
  - → "These instructions are intended to provide "back doors" or traps to implementation-specific functionality implemented in software and hardware, respectively."

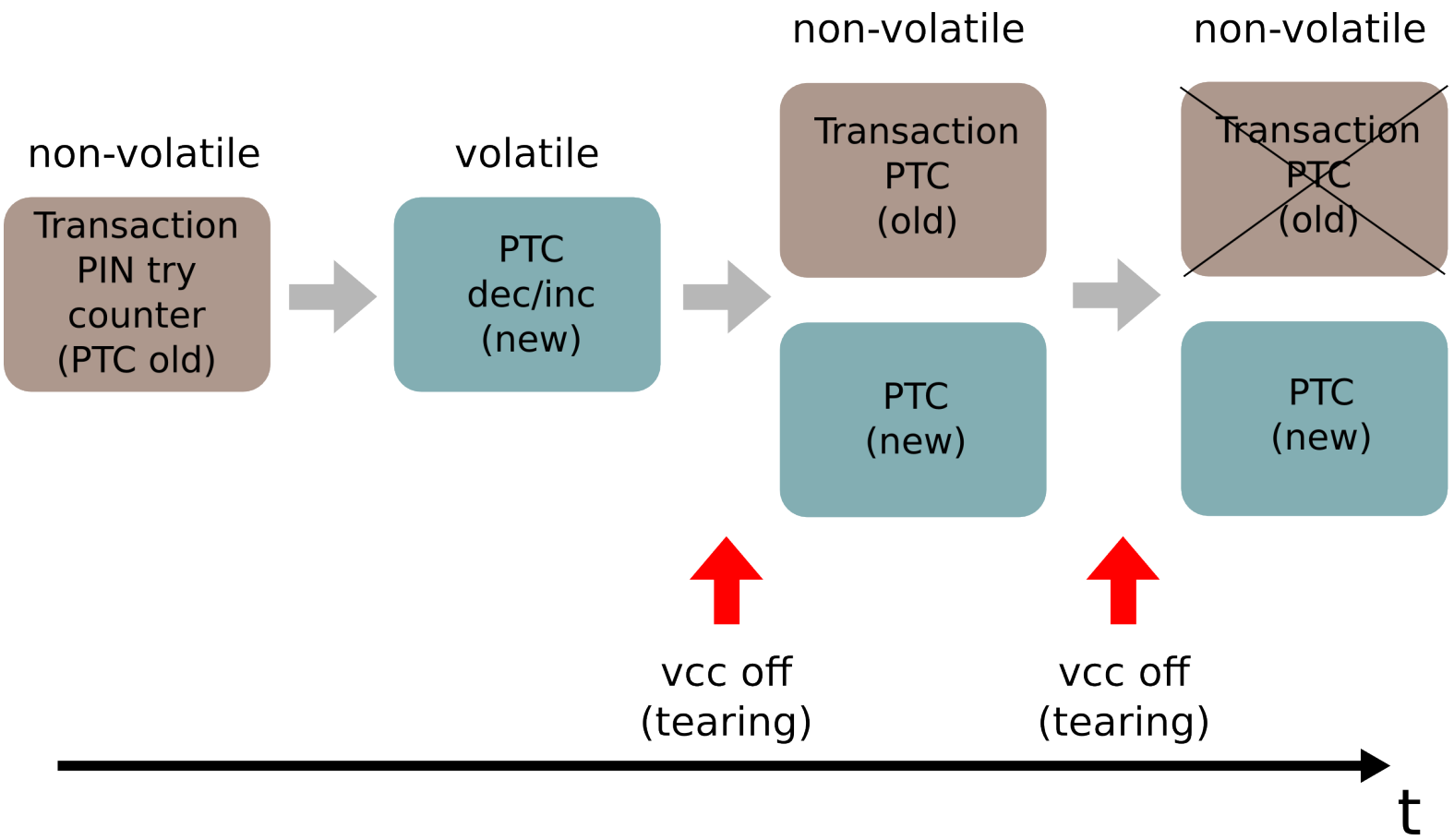
JAVA CARD FEATURES
FIREWALL



JAVA CARD FEATURES
TRANSACTION SYSTEM



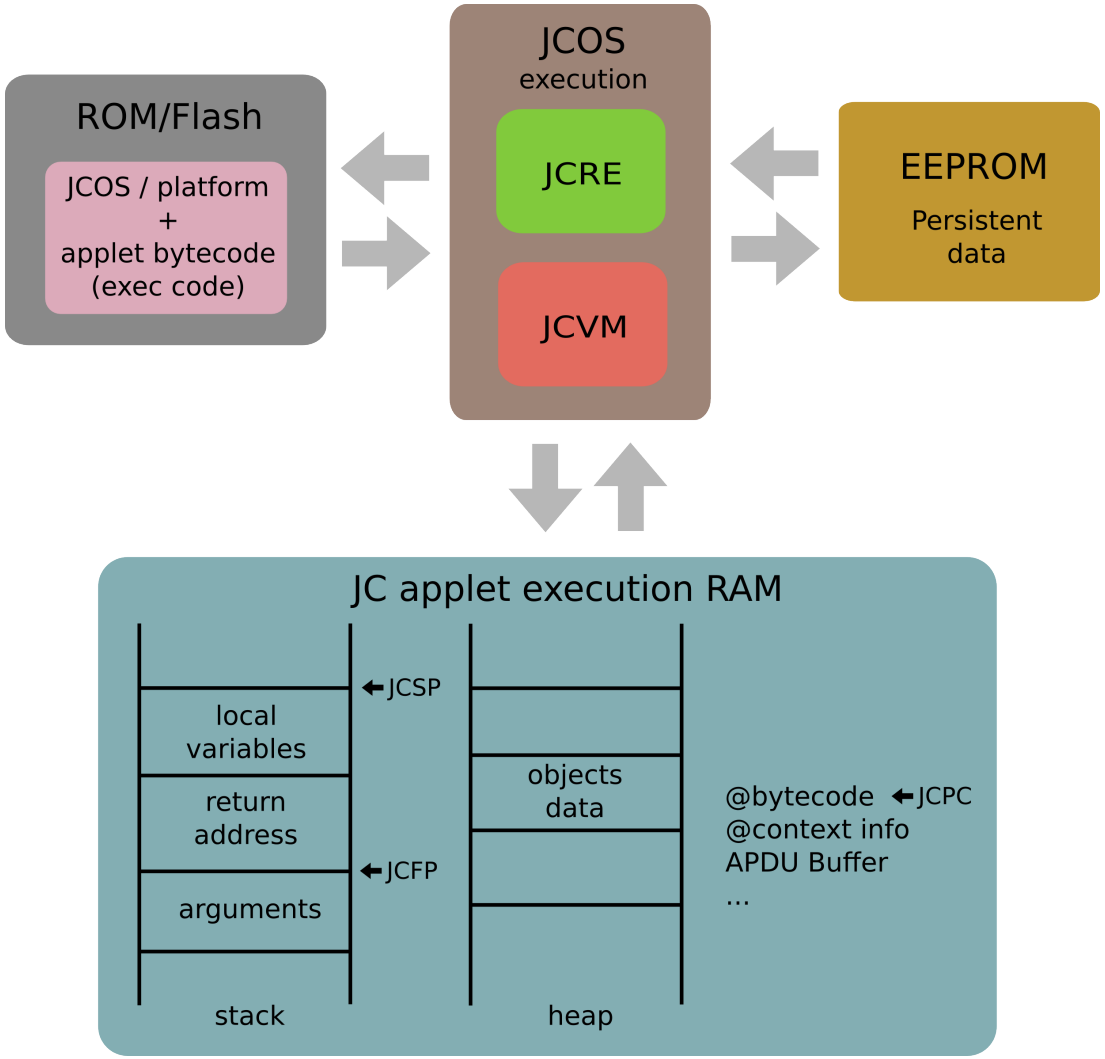
Undesired effect:



- → Type correctness.
- → No stack underflow or overflow.
- → Objects are initialized.
- → Not forged objects references on the bytecode.
- → Illegal casting of object references between classes.
- → Only allowed JC API calls.
- → Jumping in the middle of legal JC API method.
- → Jumping to data as it were code. :-)

- → symmetric encryption and decryption algorithms;
- → asymmetric encryption and decryption algorithms;
- → key interfaces;
- → signature generation and verification;
- → message digests;
- → random data generation; and
- → PIN management.

JAVA CARD FEATURES
MEMORY: ROM, EEPROM/FLASH, RAM



OVERVIEW

JAVACARD THREATS
SOGIS.ORG APPROXIMATION FOR SMARTCARDS

- → Physical attacks
- → Overcoming sensors and filters
- → Perturbation attacks
- → Retrieving keys with DFA
- → Side-channel attacks
- → Exploitation of Test features
- → Attacks on RNG
- → Ill-formed Java Card applications
- → Software attacks

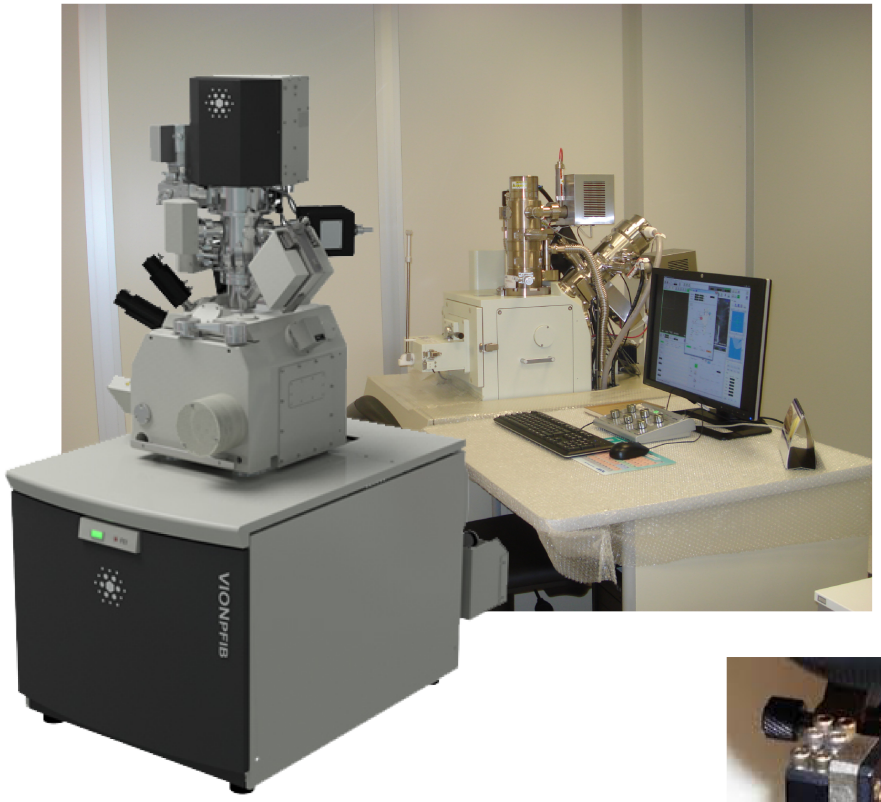
1

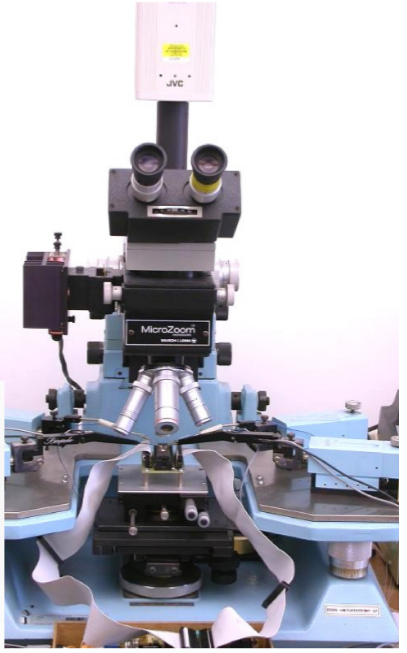
[1] http://sogis.org/

- → Physical attacks, overcoming sensors and filters, RNG and test features attacks
- → Perturbation and DFA attacks.
- → Side-channel attacks.
- → Ill-formed applets and software attacks.

PHYSICAL ATTACKS

FIB (focused-ion-beam)



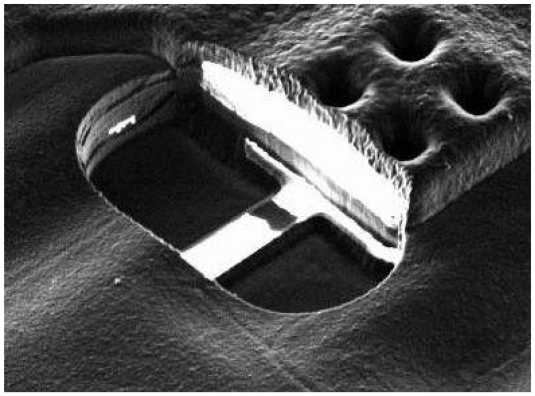


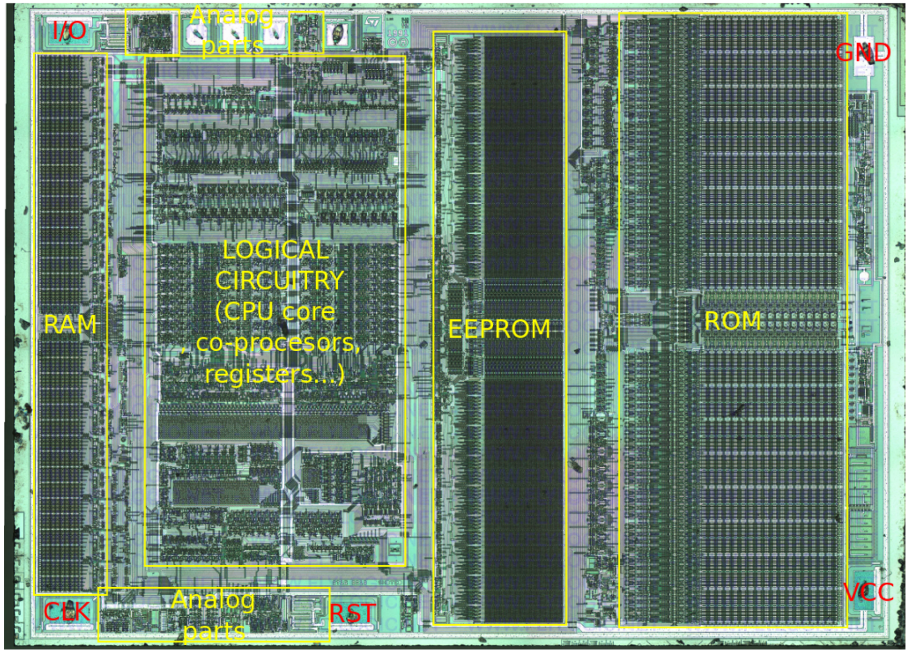
probing station

$\frac{2}{2}$

PHYSICAL ATTACKS II

Cutted semiconductor line





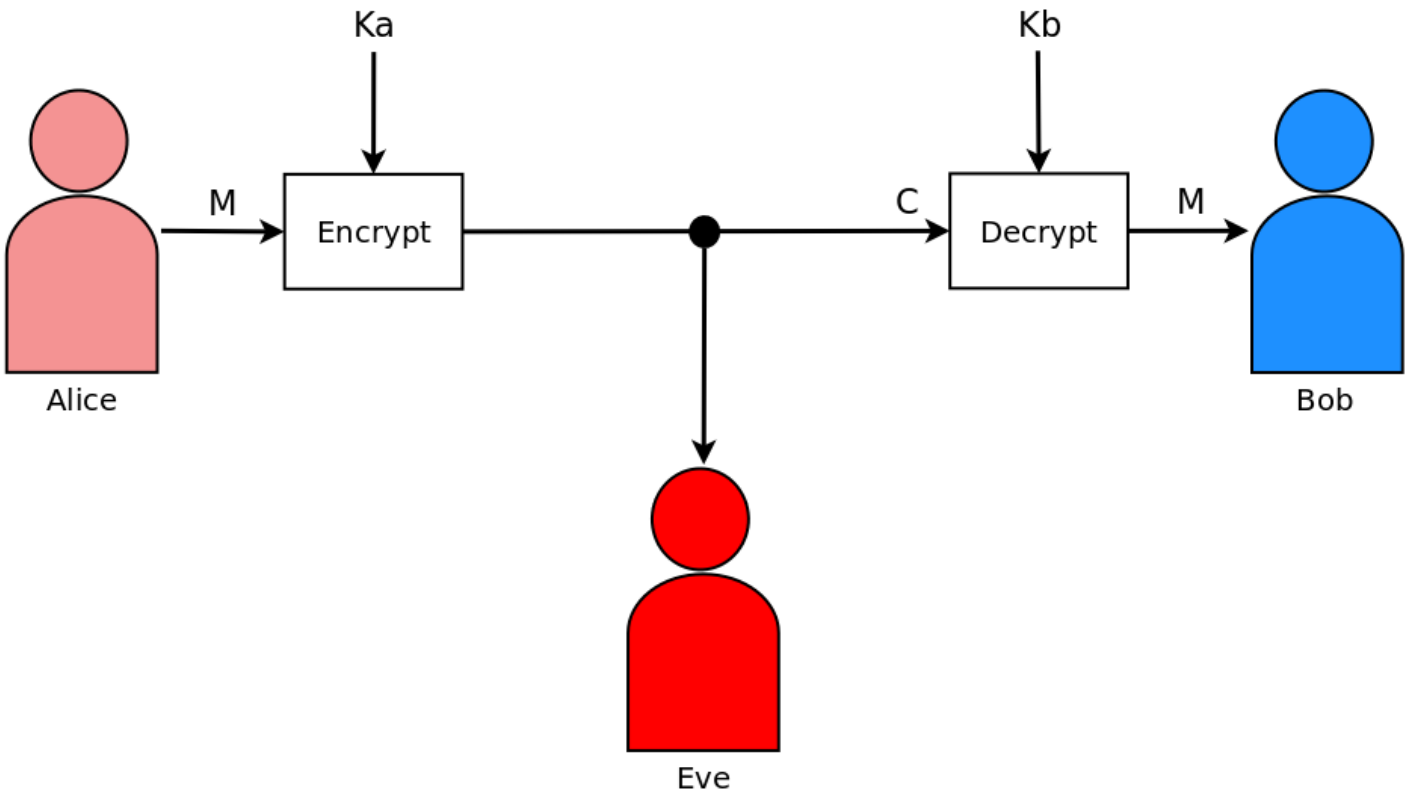
Etched microprocessor (ST16301)

[3] [4]

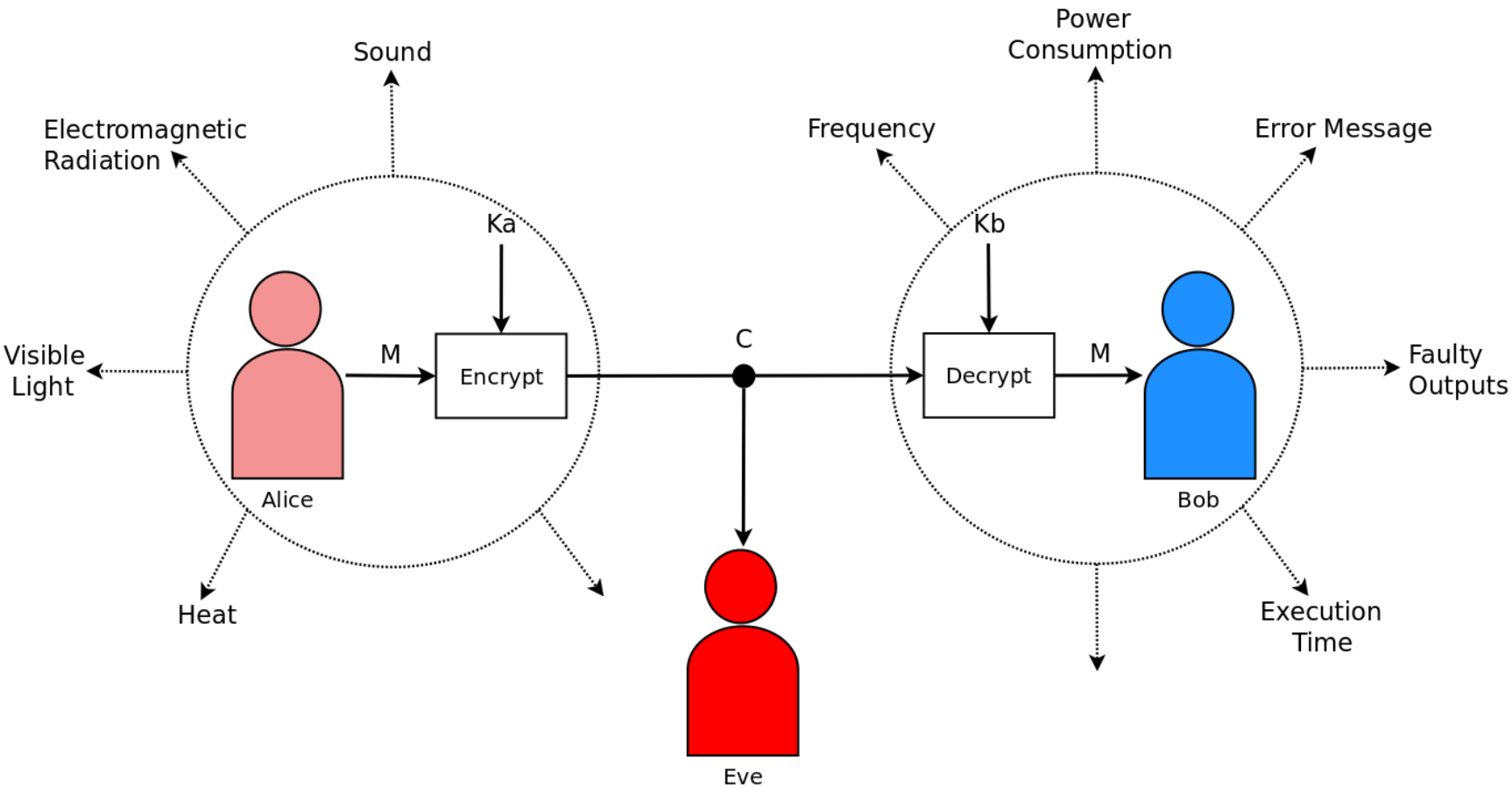
[3] https://www.usenix.org/legacy/events/smartcard99/full_papers/kommerling/kommerling_html/

[4] http://blog.ioactive.com/2007_12_01_archive.html

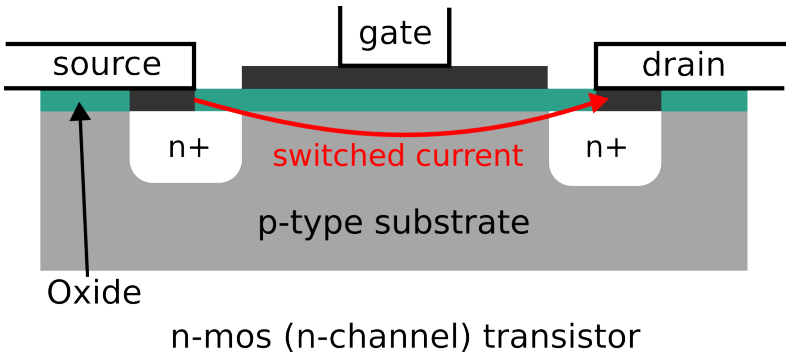
SIDE-CHANNEL ATTACKS
CLASSICAL CRYPTOANALYSIS MODEL



SIDE-CHANNEL ATTACKS
SIDE-CHANNEL MODEL



SIDE-CHANNEL ATTACKS



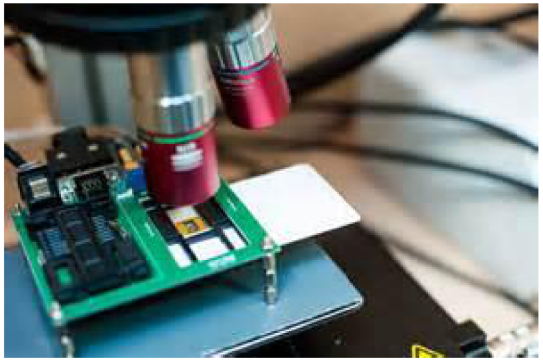
n-mos (n-channel) transistor

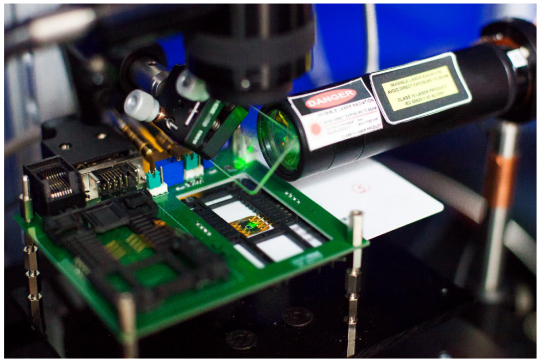
- → Current flow when charge is applied to or removed from the gate.
- → Current delivers charge to other transistors, wires and other load circuits.
- → The motion of electric charge consumes power externally detectable and observable.
- → Microprocessor logic units exhibit regular transistors switching patterns.
- → Identify macro-characteristics such as microprocessor activity.

PERTURBATION ATTACKS

- → Laser beam injects photons, which contain energy.
- → Photons are absorbed increasing its energy.
- → Temporary faults due to bit-flips.
- → May lead to code or data corruption.

Laser station





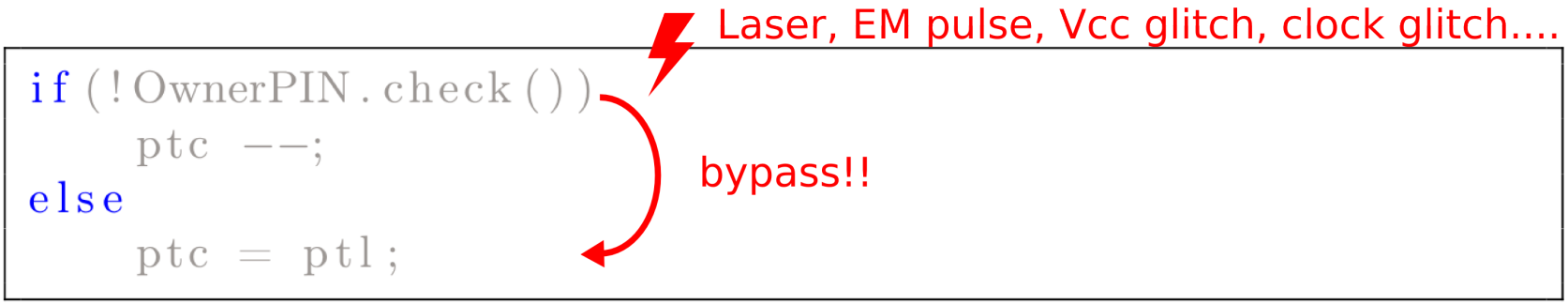
Single and double shoot

PERTURBATION ATTACKS

- Semi-invasive attack
- Change normal behavior of an IC to create exploitable errors in the operation.
- Power supply, clock, laser, EM pulse, temperature.
- Very very very powerful attack.
- Main focus on:
  - Cryptograhic algorithms
  - Instruction fetching at execution time.
- Spatial resolution.
- Backside / frontsize.

PERTURBATION ATTACKS
EXAMPLE

```
1 if (!OwnerPIN.check()) // 1 bit comparison
2     ptc --; // decrement PIN try counter
3 else
4     ptc = ptl; // reset PIN try counter
```



[5]

[5] assumption: ptc!=0

JAVA CARD ATTACKS

- → Introduce malformed codification with two objectives:
  - → Bypass Java Card firewall security checks.
  - → Modify behavior of the JCRE or JCVM.

- → How we can load malicious applets?
  - → Verified byte-code.
  - → Non-verified byte-code.

JAVA CARD ATTACKS II

- Classification and some examples: [6]
  - Bytecode verified
    - External Class Type Confusion
    - Reference persitency on transaction abortion
  - Ill-typed code
    - Simple type confusion
    - Stack underflow - overflow
    - Forge invoke static and get static bytecodes
    - Type confusion using SIO (Shared Interface Objects).

[6] There are infinite public conceptual examples in the Internet, no code at all.

- → Editing commands
- → Direct protocol attacks
- → Man in the middle
- → Replay commands
- → Bypass auth control

COMBINED ATTACKS
COMBINED ATTACKS

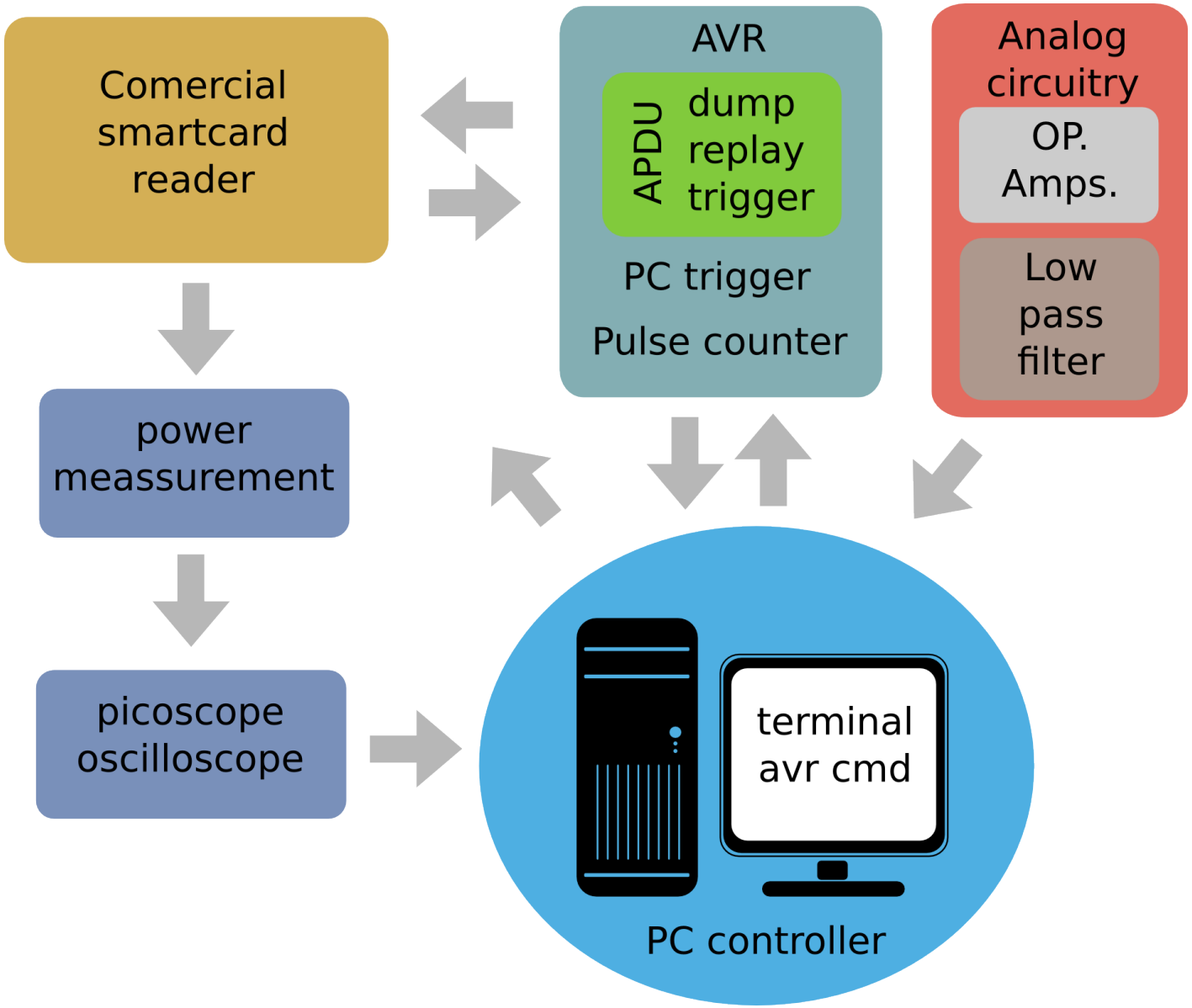
- → Those using both logic and hardware attacks
- → Most powerful attacks
- → Requires expertise and materials from different fields

OVERVIEW

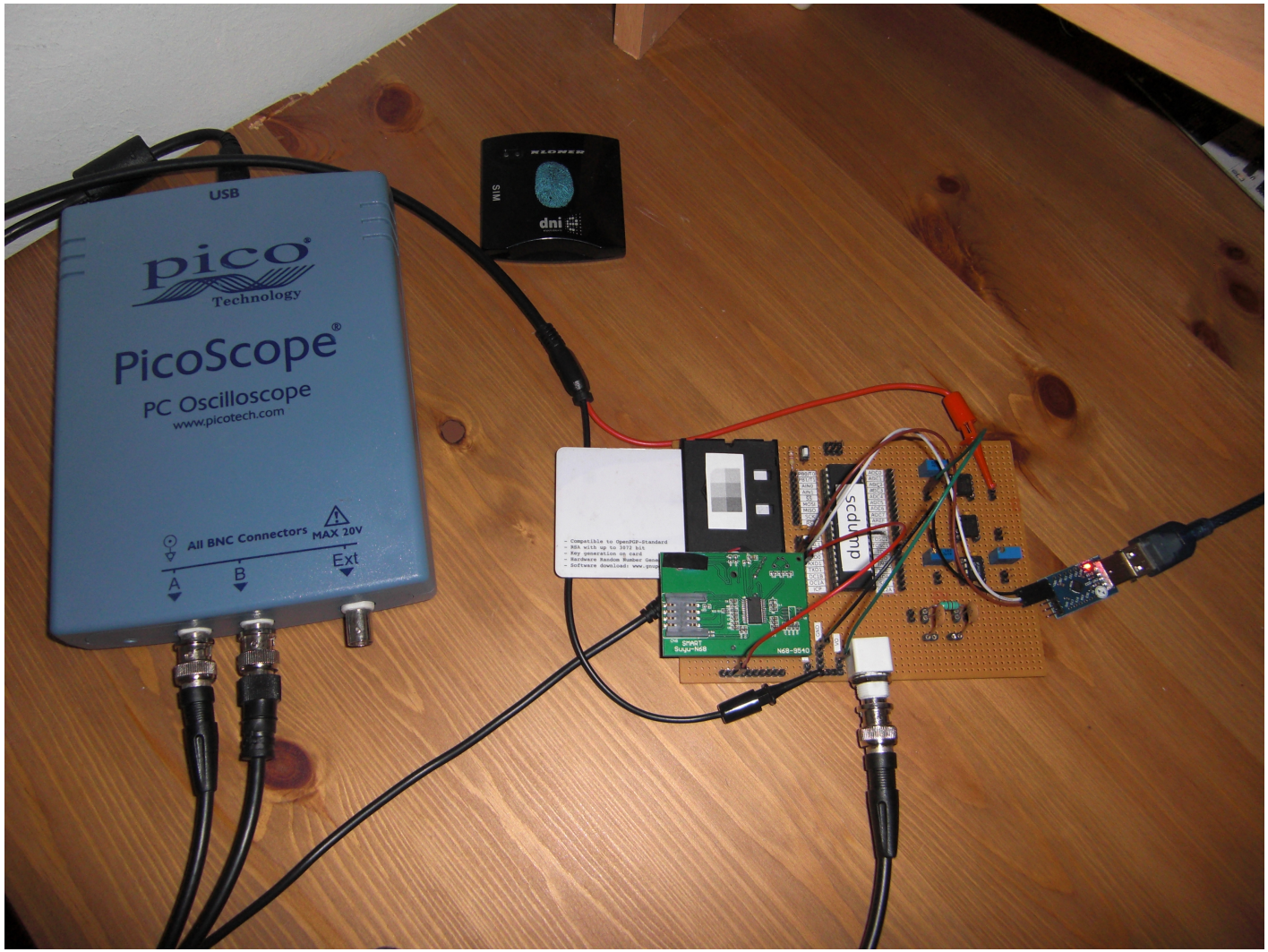
- → Oscilloscope: PicoScope 3206.
- → Home-made smart card terminal.
- → Oracle JDK / JCDK (http://www.oracle.com)
- → gpshell (http://sourceforge.net/p/globalplatform/)
- → ht editor (https://github.com/sebastianbiallas/ht)
- → scilab: (http://www.scilab.org/) [7]

[7] All the power consumption images are screen shots of this software

TEST BENCH
BOARD



TEST BENCH
MOUNTED TEST BENCH



IMPORTANT:

ALL THE SMART CARDS USED FOR THESE TESTS CAN BE BOUGHT ONLINE.
SMART CARD MODEL IS KEPT CONFIDENTIAL.

TIMING IN THE PIN COMPARISON
EXAMPLE OF TIMING IN THE PIN COMPARISON

```
1 boolean checkPin(byte[] pinData, short offset, byte length)
    {
2     for (short i = (short)0; i < length; i++) {
3         if (currentPin[i] != pinData[(short)(offset + i)]) {
4             return false;
5         }
6     }
7     return true;
8 }
```

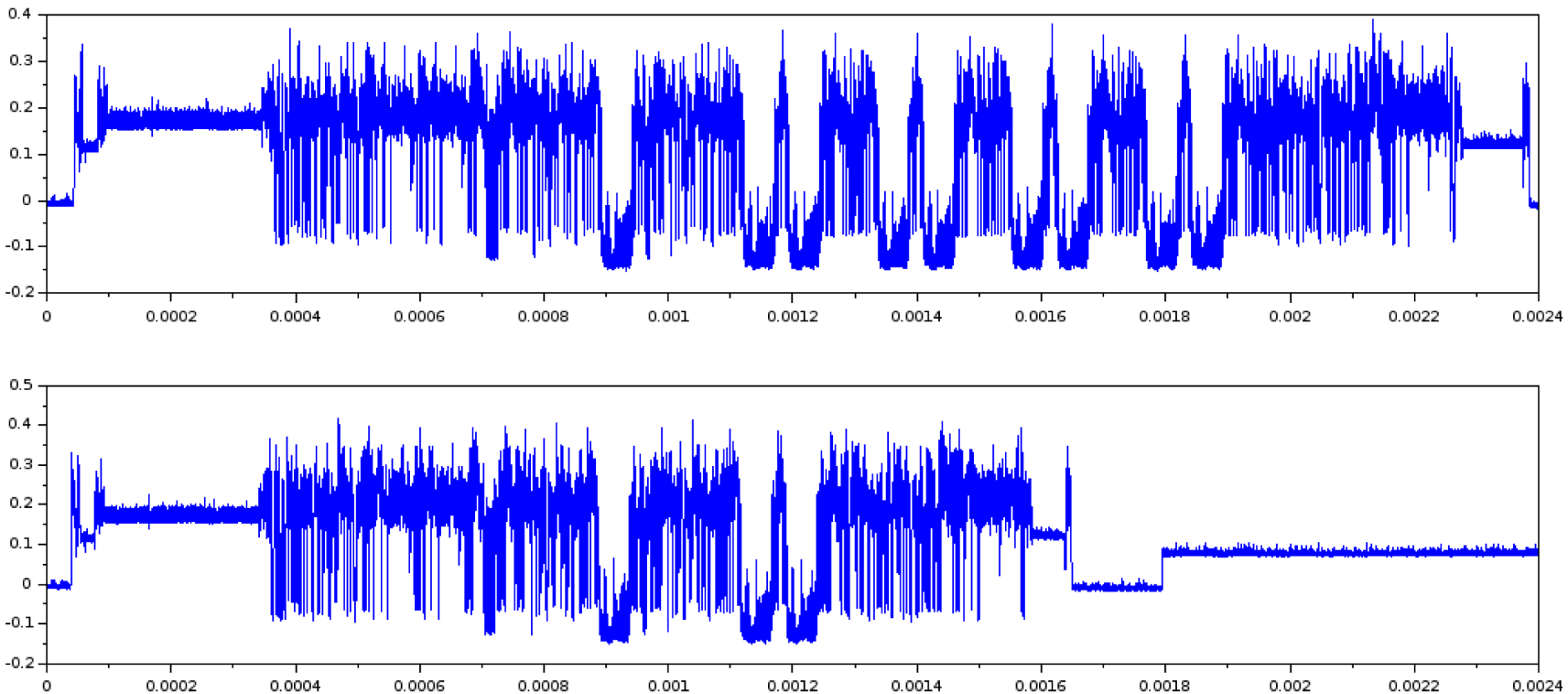
→ Applet with a bad implementation of the PIN.

→ Check all the bytes of the PIN

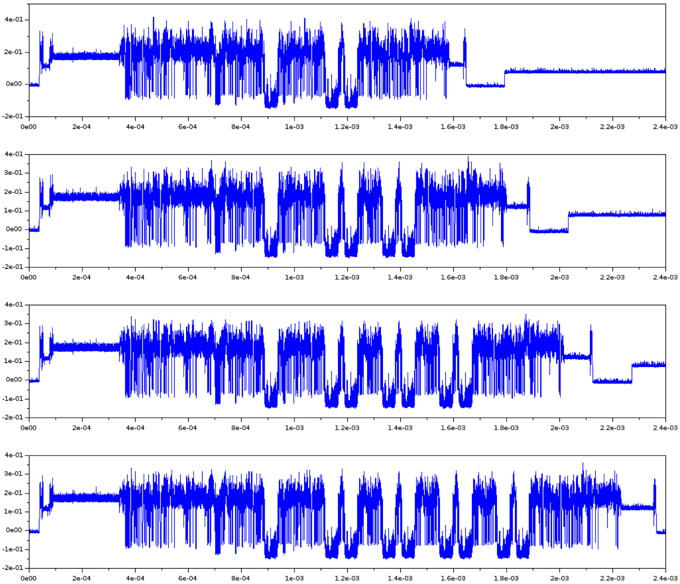
→ If bytes are not equal stop the comparison.

TIMING IN THE PIN COMPARISON

GOOD PIN VS. BAD PIN

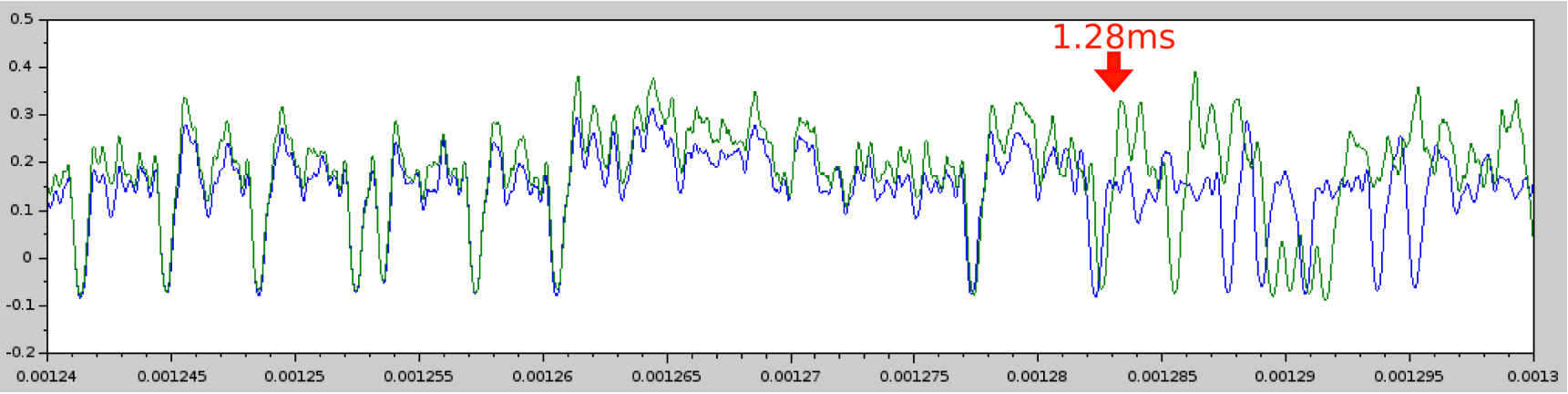


TIMING IN THE PIN COMPARISON
BAD PIN 1, 2, 3 AND 4 COMPARISON

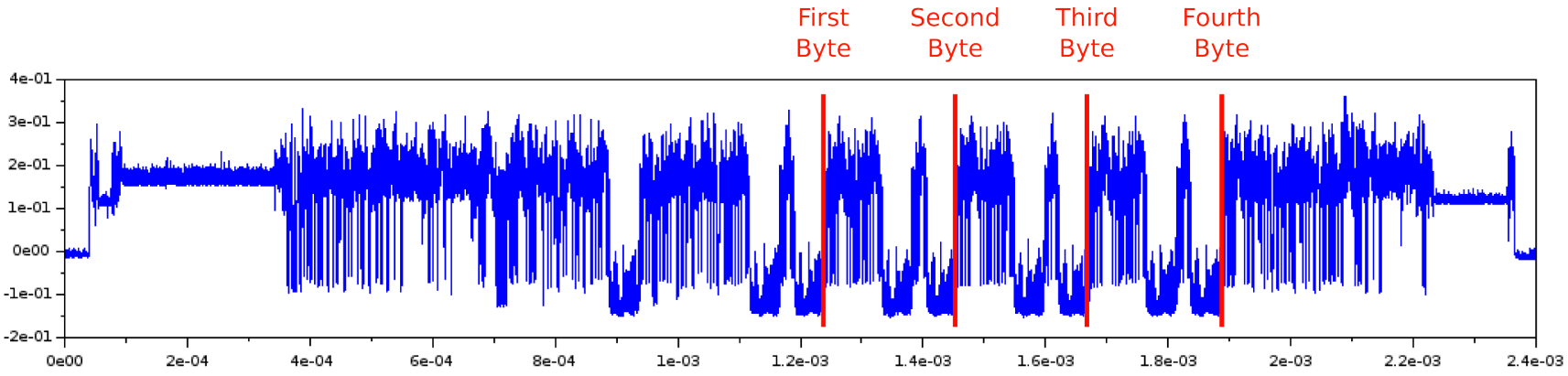


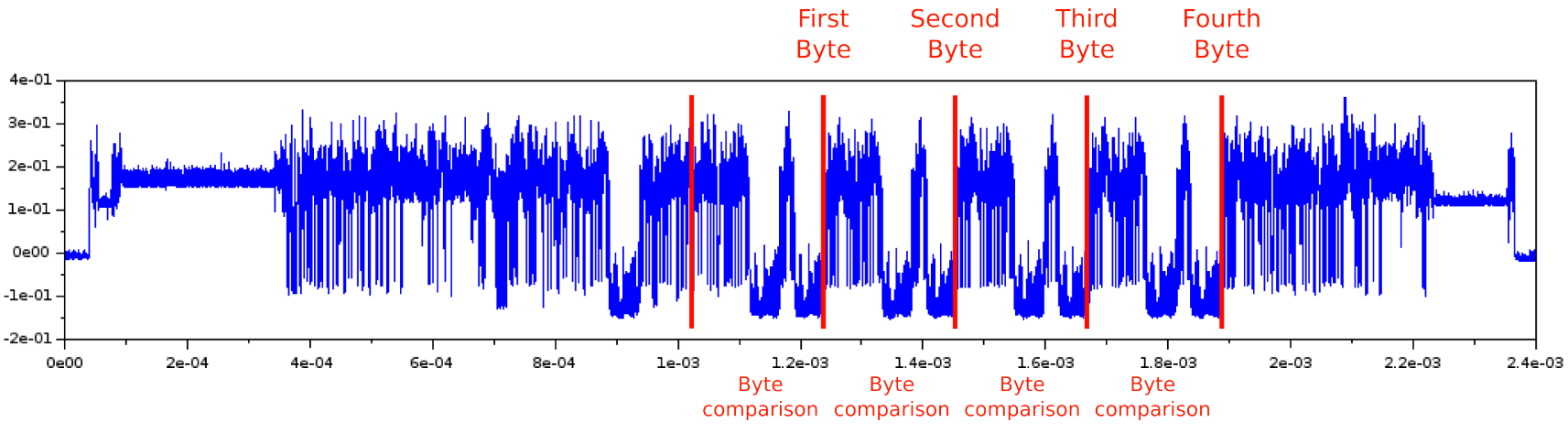
TIMING IN THE PIN COMPARISON

GOD PIN AND BAD PIN TIMING DIFFERENCES

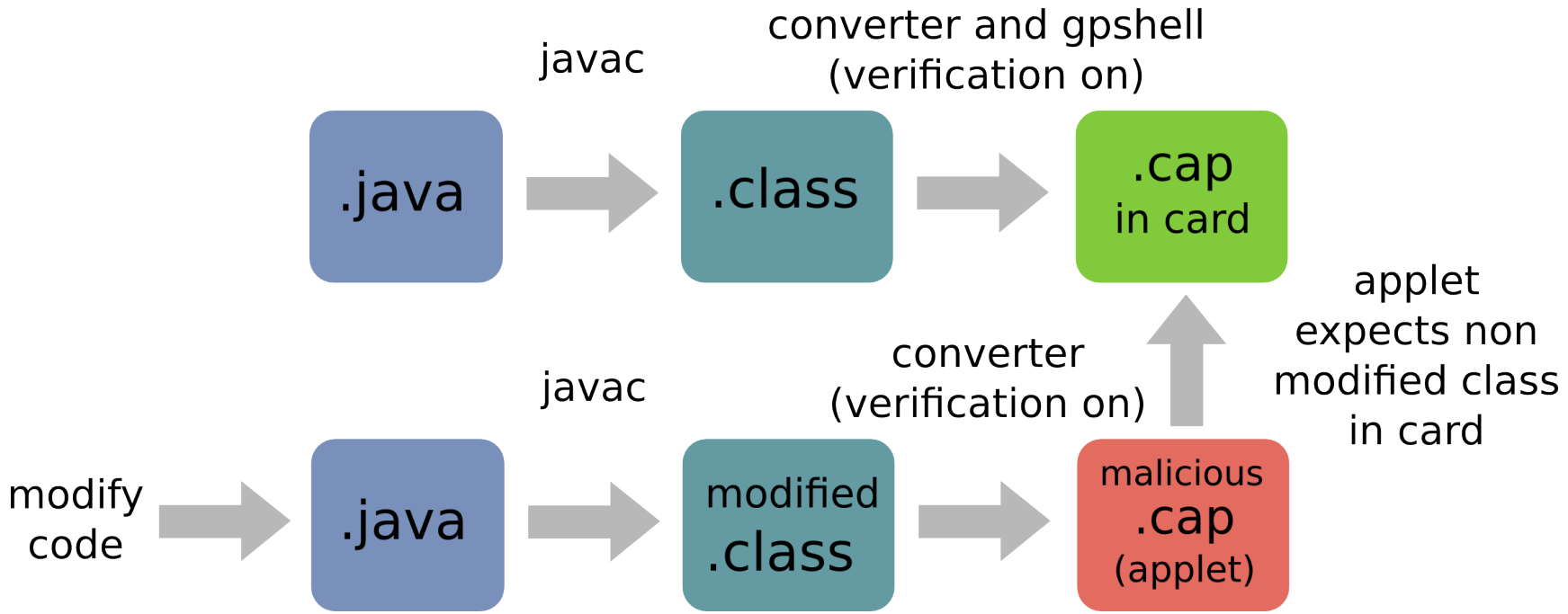


TIMING IN THE PIN COMPARISON
ANALYSIS CONCLUSIONS AND NEXT STEPS





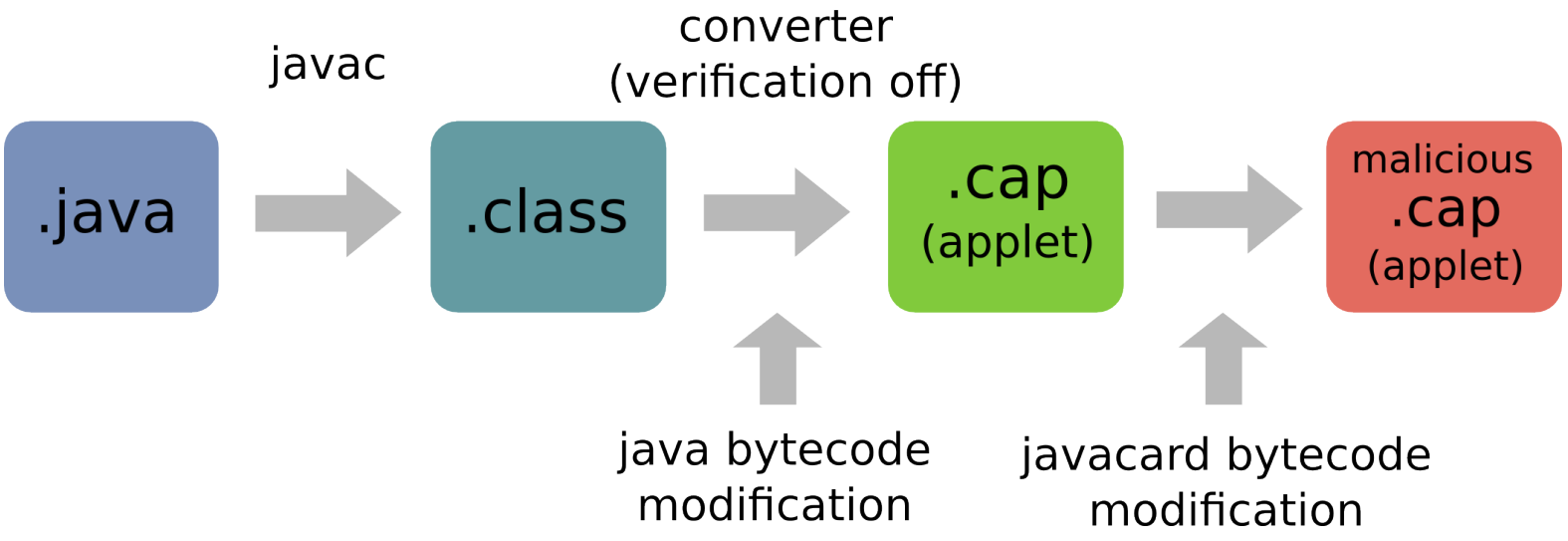
MALICIOUS APPLETS
MALICIOUS COMPILATION CONDITIONS



8

[8] http://www.researchgate.net/publication/260872448_Accessing_secure_information_using_export_file_fraudulence

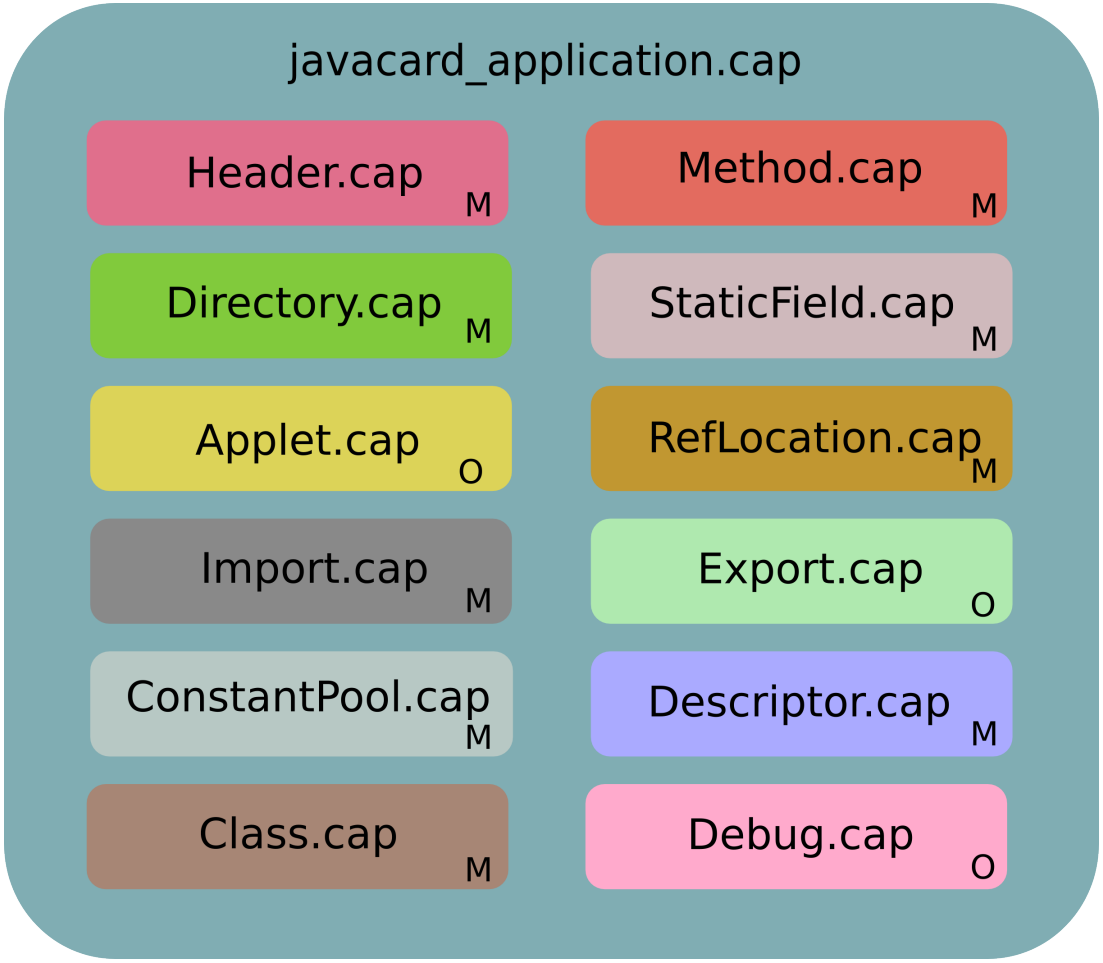
MALICIOUS APPLETS
APPLET MODIFICATION POSSIBILITITES



9

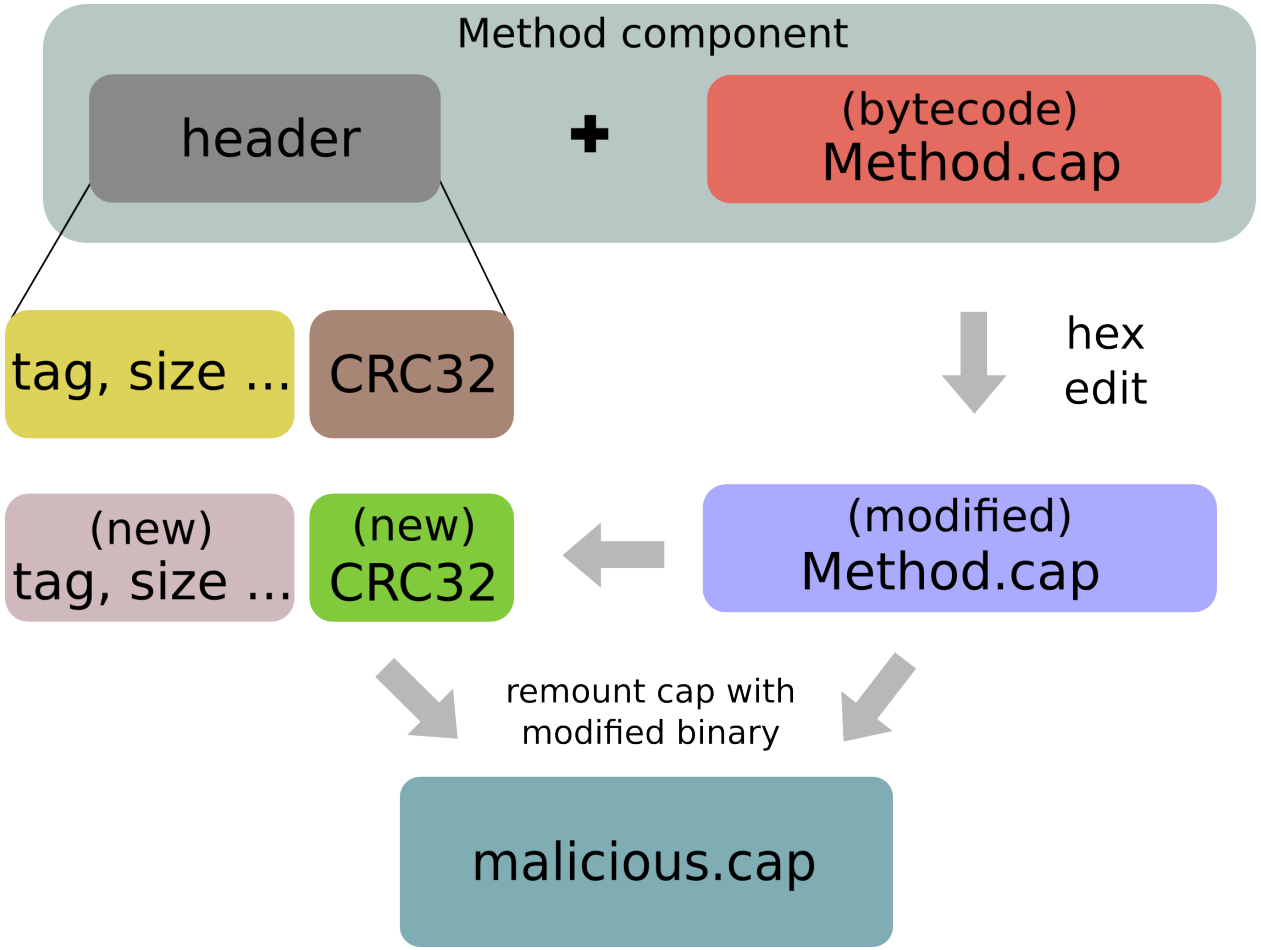
[9] http://www.cs.ru.nl/E.Poll/papers/cardis08.pdf

CAP FILE MANIPULATION
CAP CONTENT



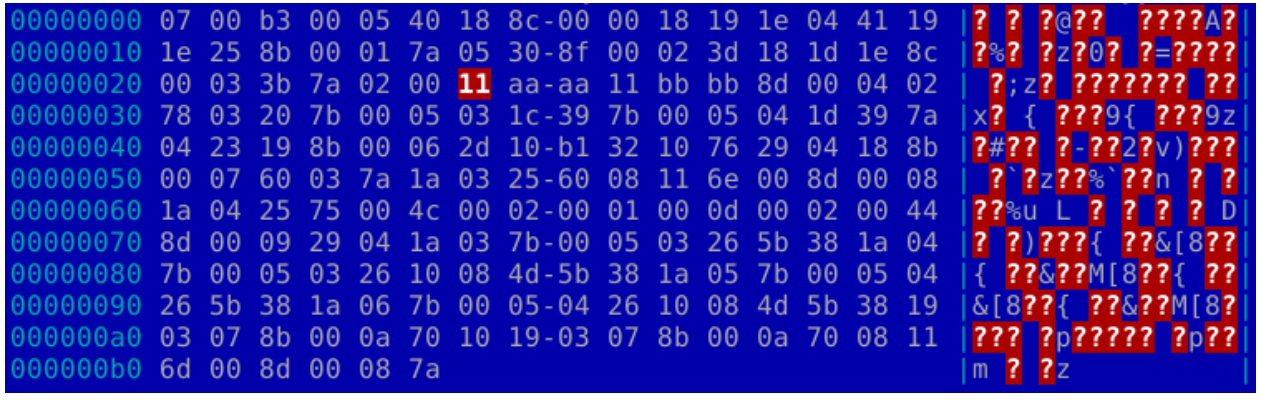
[10] [10]M - Mandatory; O - Optional

CAP FILE MANIPULATION
METHOD MODIFICATION

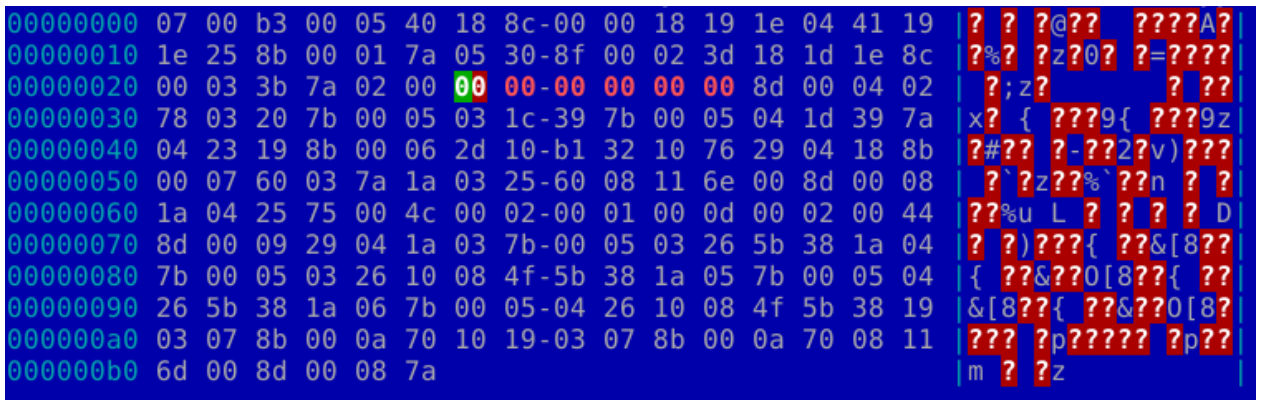


CAP FILE MANIPULATION
BINARY MODIFICATION WITH HT EDITOR

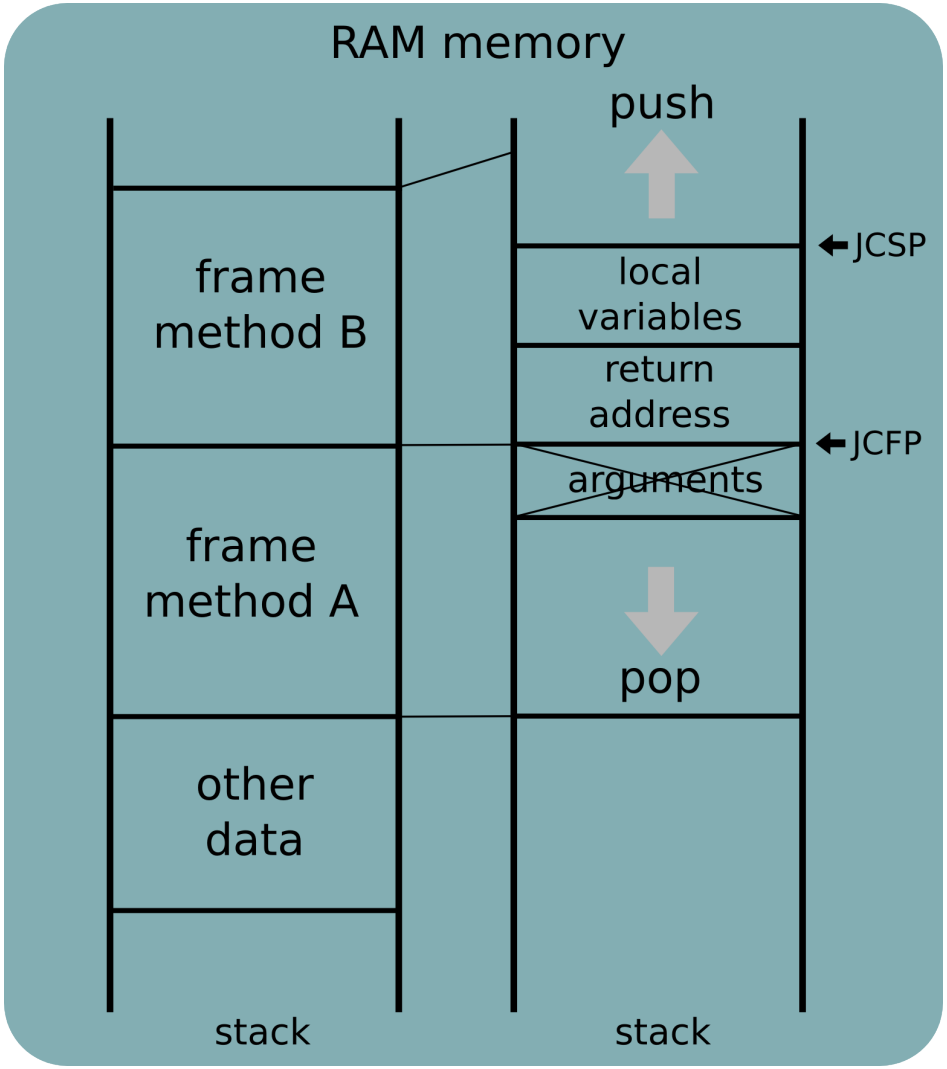
Original Method.cap



Underflow modified Method.cap



CAP FILE MANIPULATION
UNDERFLOW IN THE JC STACK EFFECT



MALICIOUS APPLETS
ATTACK RESUTLS

Attack	Cref	Card A	Card B
Simpe type confusion	...	Fail	Fail
External class mod	Success	Success	...
A. Transaction Simple	...	Success	...
A. Transaction Fault	...	Success	...
Underflow	...	Fail	...

THE END MY FRIEND!

Any Question?
Thanks all!

find us at www.bitwise.cat