FTF 2016
TECHNOLOGY FORUM

# SMARTCARD SECURITY: COUNTERING SIDE CHANNEL & FAULT ATTACKS

## FTF-DES-N1892

DR. MARTIN SCHAFFER | SENIOR PRINCIPAL
HEAD OF SECURITY MATURITY & CERTIFICATION
FTF-DES-N1892
MAY 17, 2016

PUBLIC USE

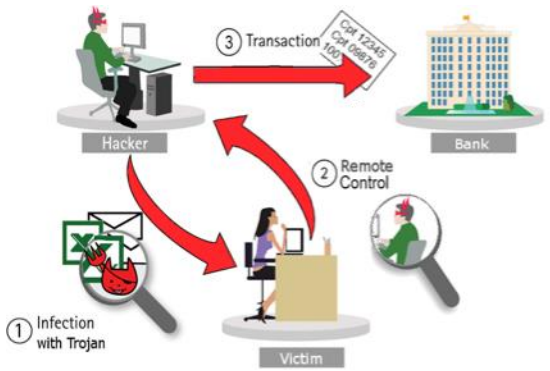
# AGENDA

- Physical Attacks
- Security Evaluation & Certification
- JHAS Attack Rating
- Summary

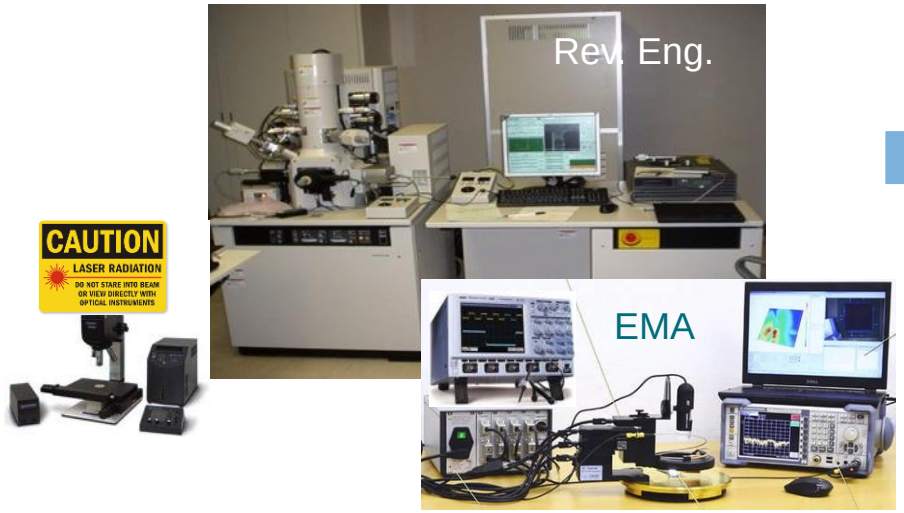
# PHYSICAL ATTACKS

# Functional vs. Physical Security

## Remote / Online / Logical Attacks




## Local / Offline / Physical Attacks




Fault Injection

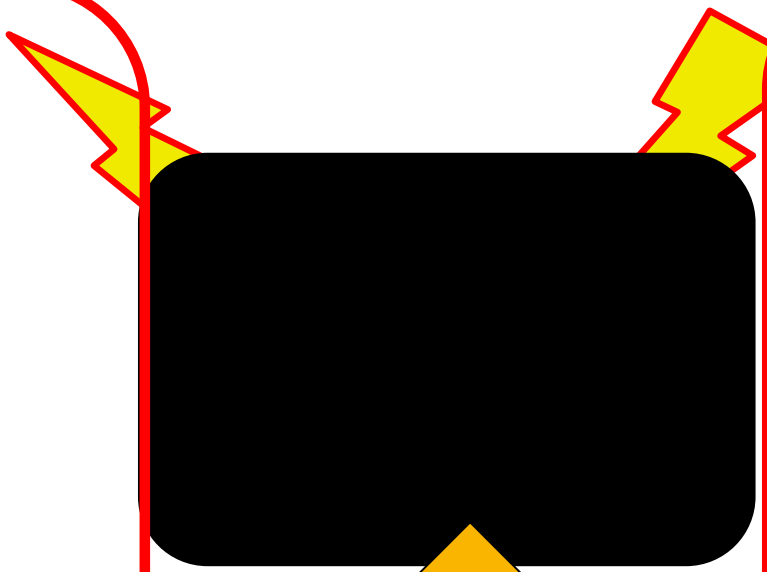
## Functional Security

- Standard crypto toolbox
- SW-only is good enough
- SW firewall
- Virtualization techniques
- Maybe HW accelerators, e.g. crypto copros, TZ,…

## Physical Security

- Functional Security
- Implementation Security
  - Side-channel (e.g., EMA), Fault Injection, Reverse Engineering
- Secure Element
- HW firewall, HW trust anchor
- HW-SW co-design

# Attacks on Smart Cards and Similar Devices



**Side-Channels**
Power Consumption
Electromagnetic Emanation
Timing
Communication (Errors)
Heat Emanation

**SCA-Attacks**
Simple Power Analysis (SPA)
Differential Power Analysis (DPA)
Template Attacks
Timing Analysis

**Non Invasive Attacks**

What is implemented?
**Reverse Engineering**

**Fault Injection**
Laser Beam
Power- Clock Glitches
Probes
EM Pulses
Focused Ion Beam
Environmental Stress

**Fault Attacks**
Attacks on Algorithm
Attacks on Programm Flow
Single Bit vs. Multiple Bit
Differential Fault Attacks

**Semi- / Invasive Attacks**

# SIDE-CHANNEL ANALYSIS ATTACKS

# Pentagon Pizza Attack

## Washington Domino Pizza Index (early ’90)

**Pentagon, at any normal evening about 12 - 15 pizzas…**

**But every once in a while 36 - 45 pizzas…**

**Panama, Gulf War, …**

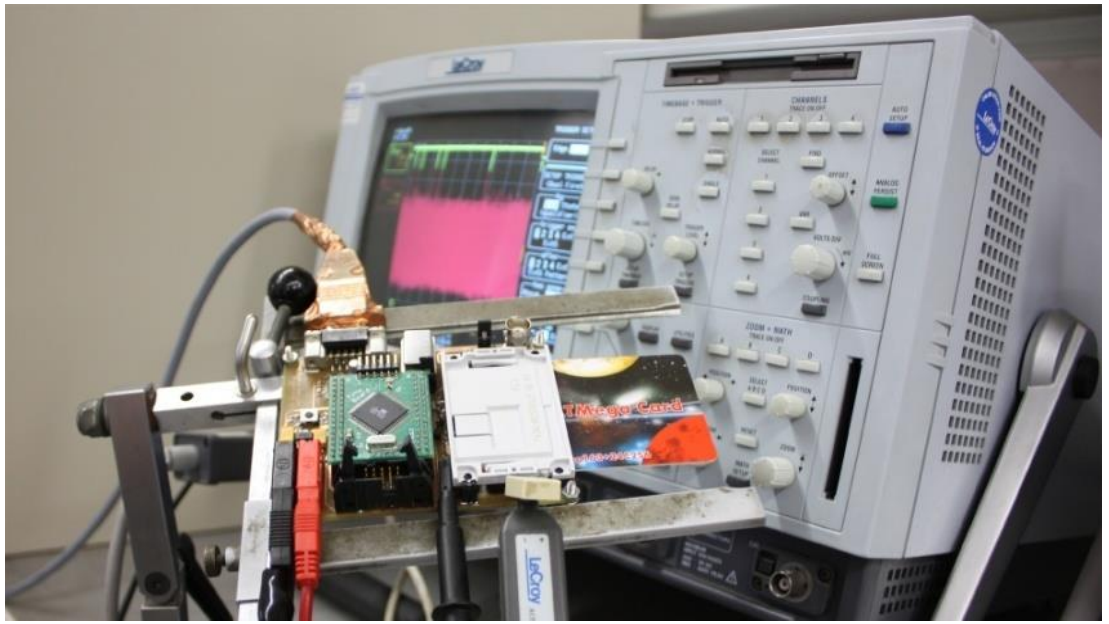
**Information leakage without any official press announcement!**

**Power**
**SPA: Simple Pizza Analysis**

**Power**
**DPA: Differential Pizza Analysis**

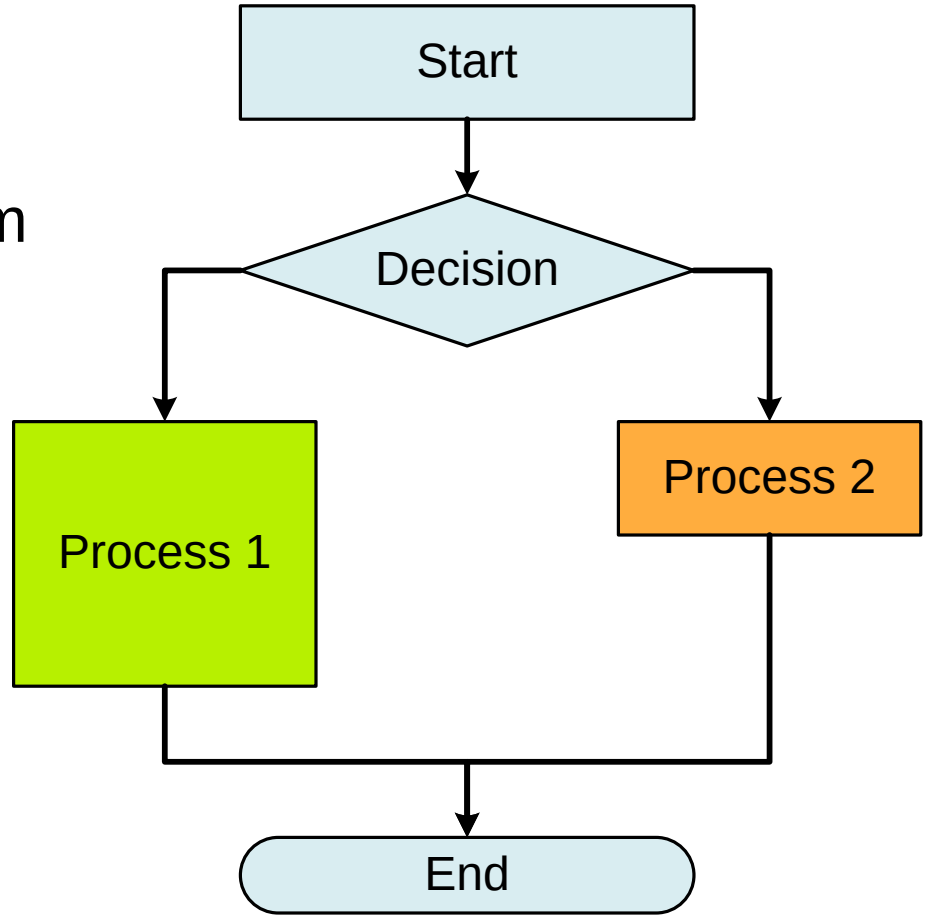
# Side-Channel Analysis Attacks

- Timing Attacks
  - Measure time for calculation → statistically extract secret
- Power Measurement Attacks
  - Measure power consumption during calculation → extract secret
- Electromagnetic Emanation Attacks
  - Same principal as power attacks
  - Very local measurements possible
- Communication
  - Analyze the response to commands
- Heat measurements
  - Can be used to localize crypto units on a chip



# Timing Attacks

- Processing Time depends on Secret Data
- Get Secret by measuring runtime of program



# PIN Code Verification

- 8 Digit PIN code used to grant data access
- Entering one PIN takes minute (card restricts higher frequency)
- Brute Force Attack:

$$\frac{10^8}{60 \cdot 24 \cdot 365} \approx 190 \; years$$

# PIN Code Verification

- Straightforward Implementation of PIN verification:

```
for i=0 to 7
{
  if PIN[i] != userInput[i]
    return error;
}
return ok;
```

- Possible Attack:
  - Try all 10 possible values for 1st digit and measure execution time
  - Execution with correct value will take longer
  - Take correct 1st digit and continue with 2nd one

$$8 \cdot 10 = 80 \text{ min}$$

# PIN Code Verification

- Countermeasure: Add random delay at end of comparison

```
retVal = ok;
for i=0 to 7{
  if PIN[i] != userInput[i]{
    retVal = error;
    break;
  }
}
wait random time;
return retVal;
```

# PIN Code Verification

- Attack still possible:
  - Try all 10 possible values for 1st digit and measure execution time many times
  - Take mean execution time for each value
  - For the correct digit the mean execution time will be longer
- Number of measurements per digit depends on length of break

$$\frac{32 \cdot 8 \cdot 10}{60} \approx 43\,hrs$$

# PIN Code Verification

- Best countermeasure is timing invariant code
- Code should have no conditional branch
- If conditional branch cannot be avoided:
  - Execution time of both branches equal
  - Be aware the execution time of a branch command can depend on the fact that a branch is taken or not

# PIN Code Verification

- Code resistant against timing attacks:

```
check = 0;
for i=0 to 7
{
  check |= PIN[i] ^ userInput[i];
}
if (check == 0){
  return ok;
} else {
  return error;
}
```

# Timing / SPA Attack: Example RSA

See also Livedemo at Booth 331 from Axel Poschmann

- Modular exponentiation: $x = m^k \bmod N$

$x = m$

for $i = n-2$ down to $0$

execution time depends on value of secret bit $k_i$!

$x = x^2$

if $(k_i == 1)$ then

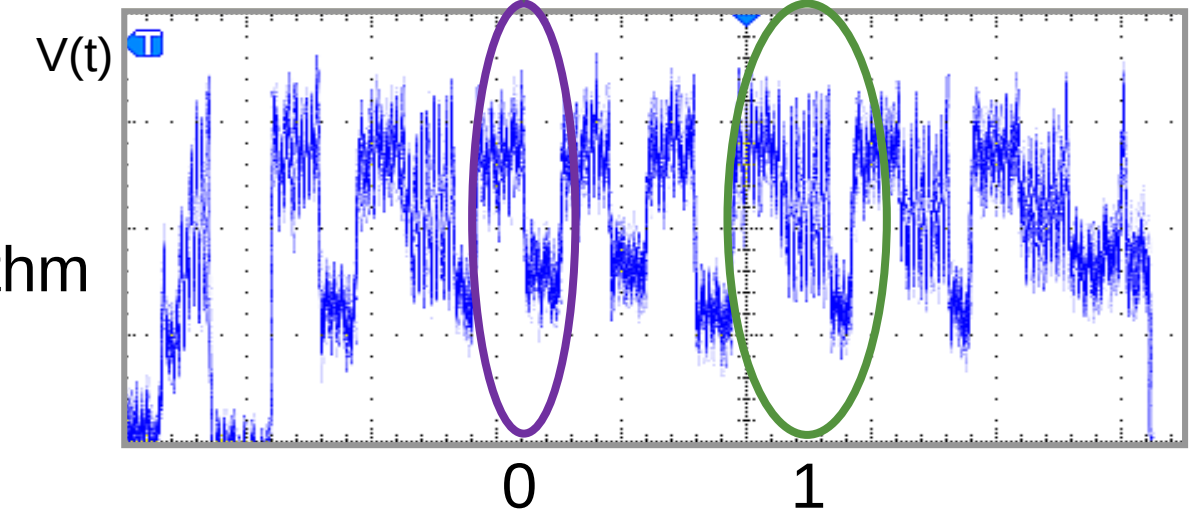
$x = x \cdot m$

endfor

return $x$

n … number of exponent bits

square & multiply algorithm



# Good Practice to Avoid Information Leakage

- Unpredictable timing
- Unpredictable program execution order
- Where possible clear the local stack before returning from a function
- Add randomization where possible
- Avoid any branches on secret data
- Clear local variables directly when no longer needed

# FAULT ATTACKS

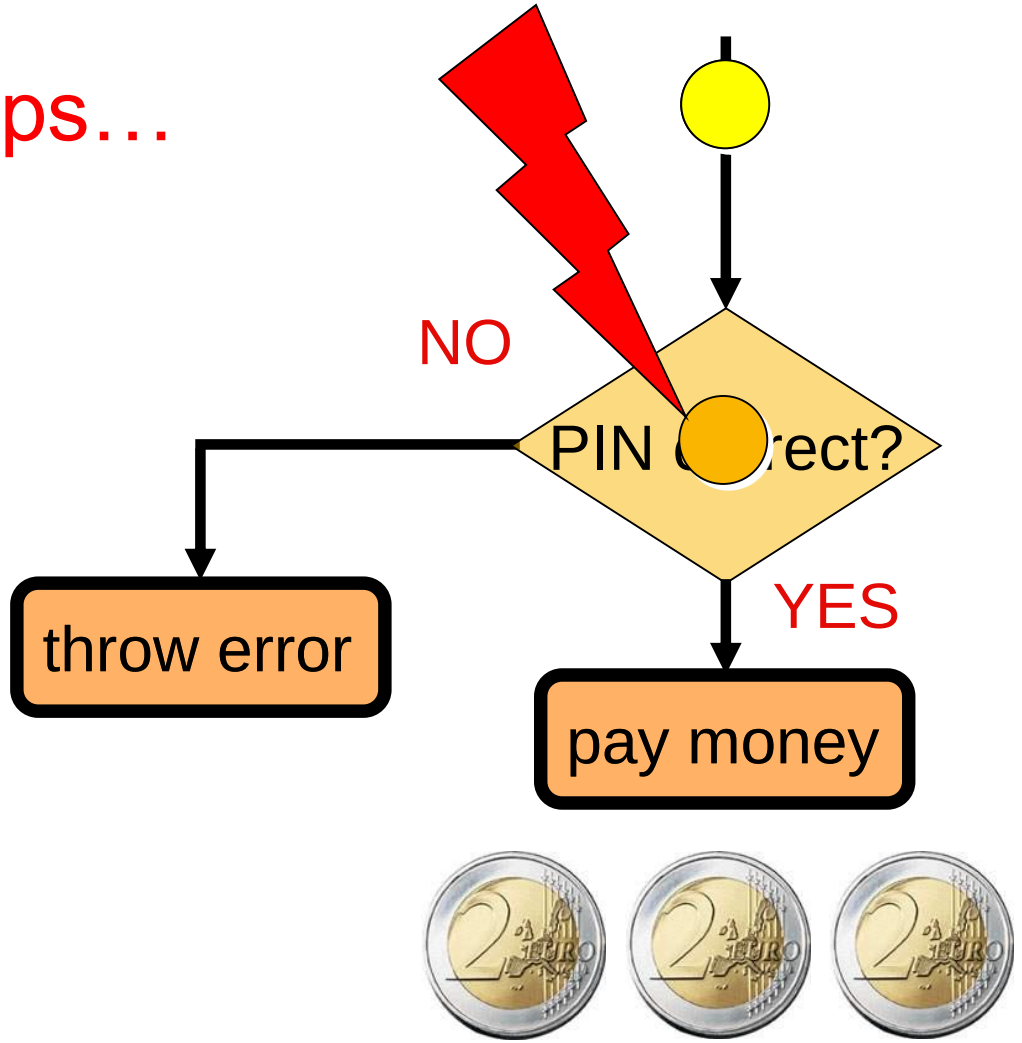
# Fault Attacks

- Introduce a fault in the computation
  - Laser beam
  - Flash light
  - Voltage burst
  - Clock jitter
  - Permanent fault (Focused Ion Beam)
- Use the faulty output as additional information
  - AES skip last key add
- Manipulate branch
  - IO loops, conditional branches
- Change security setting

Riscure laser system (diode laser)

# Simple Fault Attacks: Code Execution

attack critical jumps…

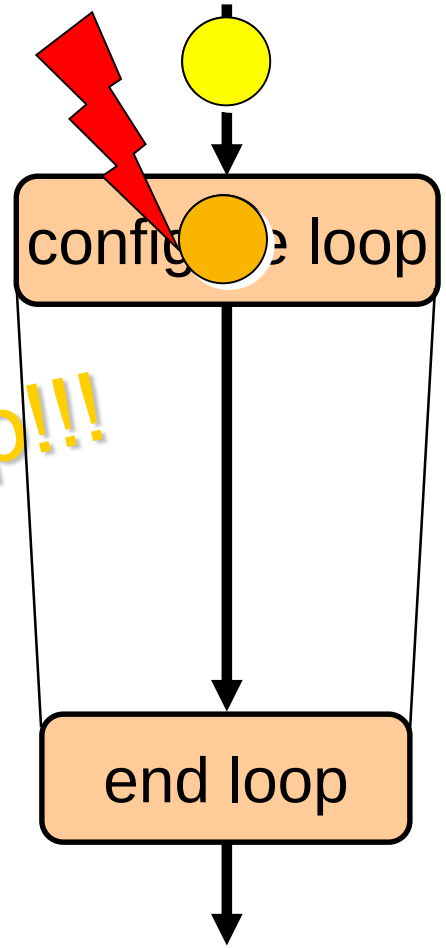





19

# Simple Fault Attacks: Code Execution

attack I/O loop…
(copy from buffer)




There is always an I/O loop!!!

Hello world!

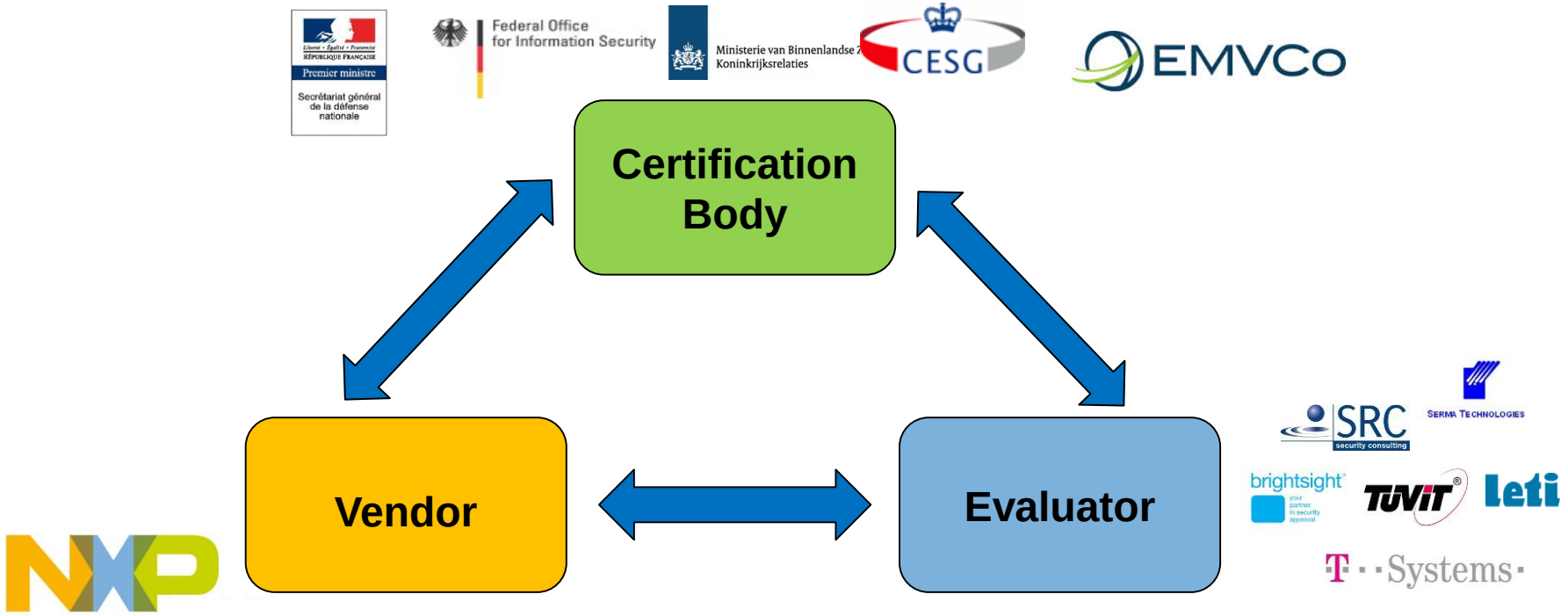
Memory dump: FB AE 04 23 82 F3 D1 …

# Good Practice to Protect Flow Control / Data Integrity

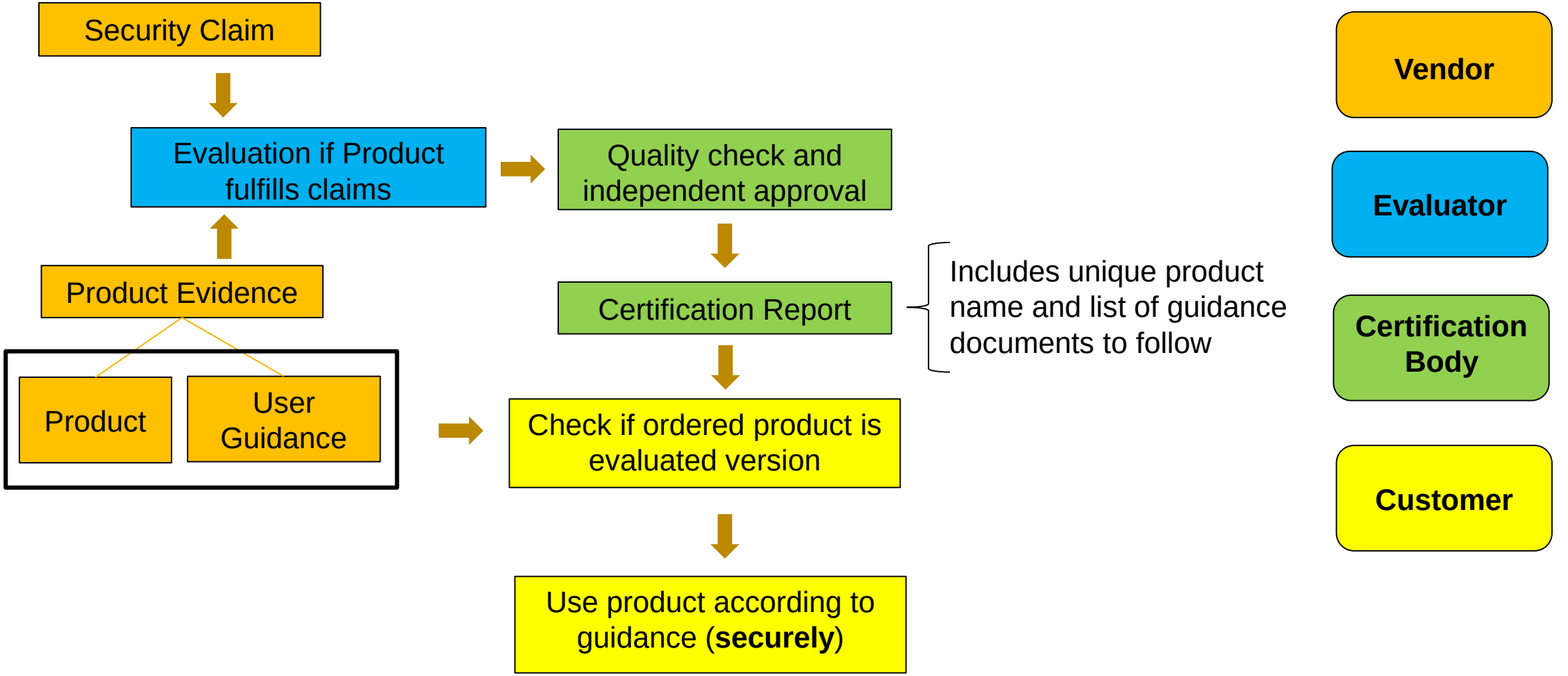
- Program flow verification
- Dummy operations, beware to choose these wisely!
- Avoid FF and 00 as security values, use values with inverse, e.g. A5 and 5A.
- Use complex patterns for decision information (not only a single bit)
- Checks at the beginning of each function (in function)
- Checks before function return (in function) and after function return (in caller)
- Check and double check
- Double executions, where the second is the inverse of the first evaluation
- Write secure code preferably in assembly (compile may remove redundancy)
- Check data (integrity) before use, NOT after!
- Make sure that checks are always performed
- Check after write

# SECURITY EVALUATION & CERTIFICATION

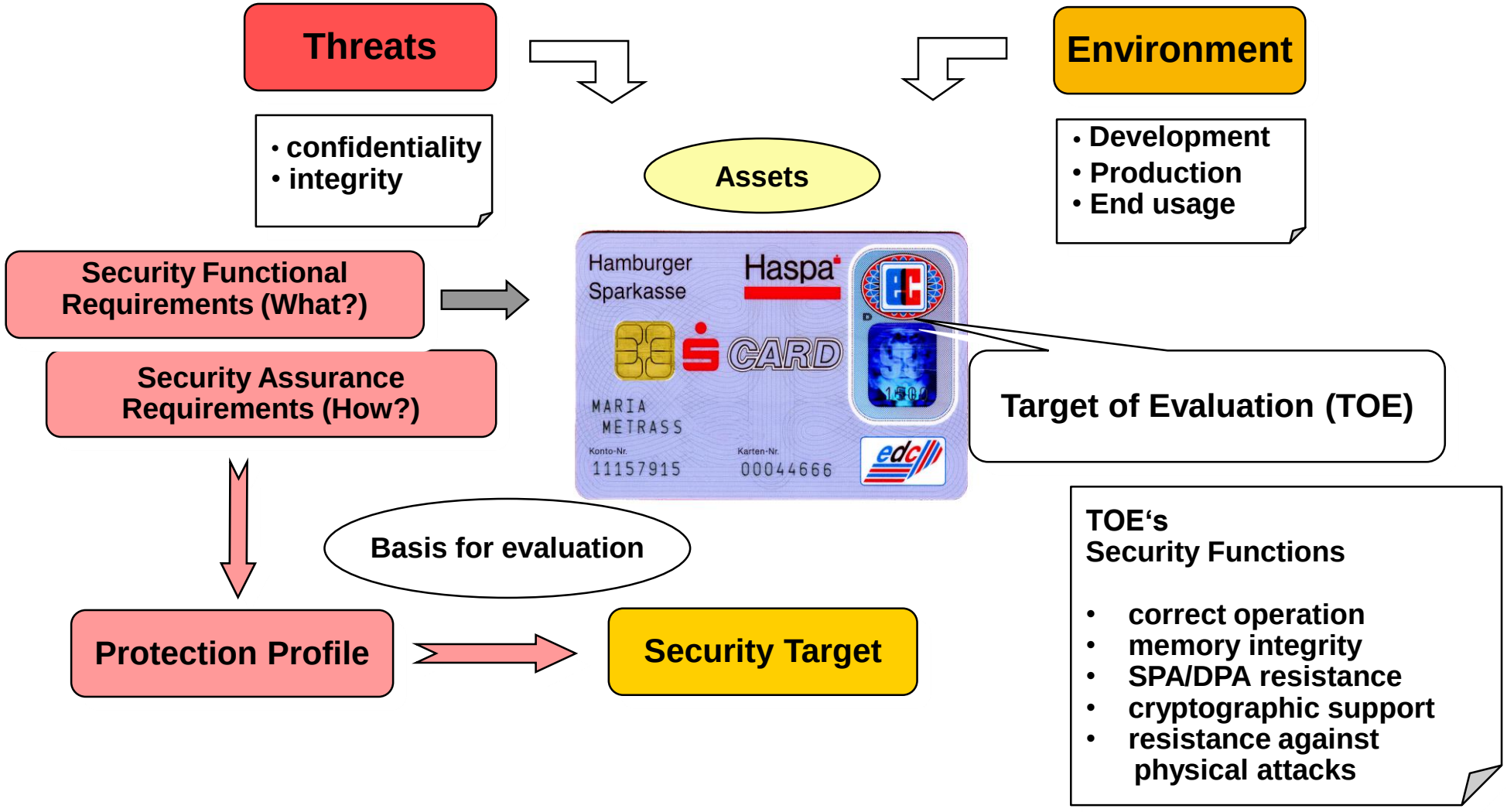
# Security Evaluation & Certification



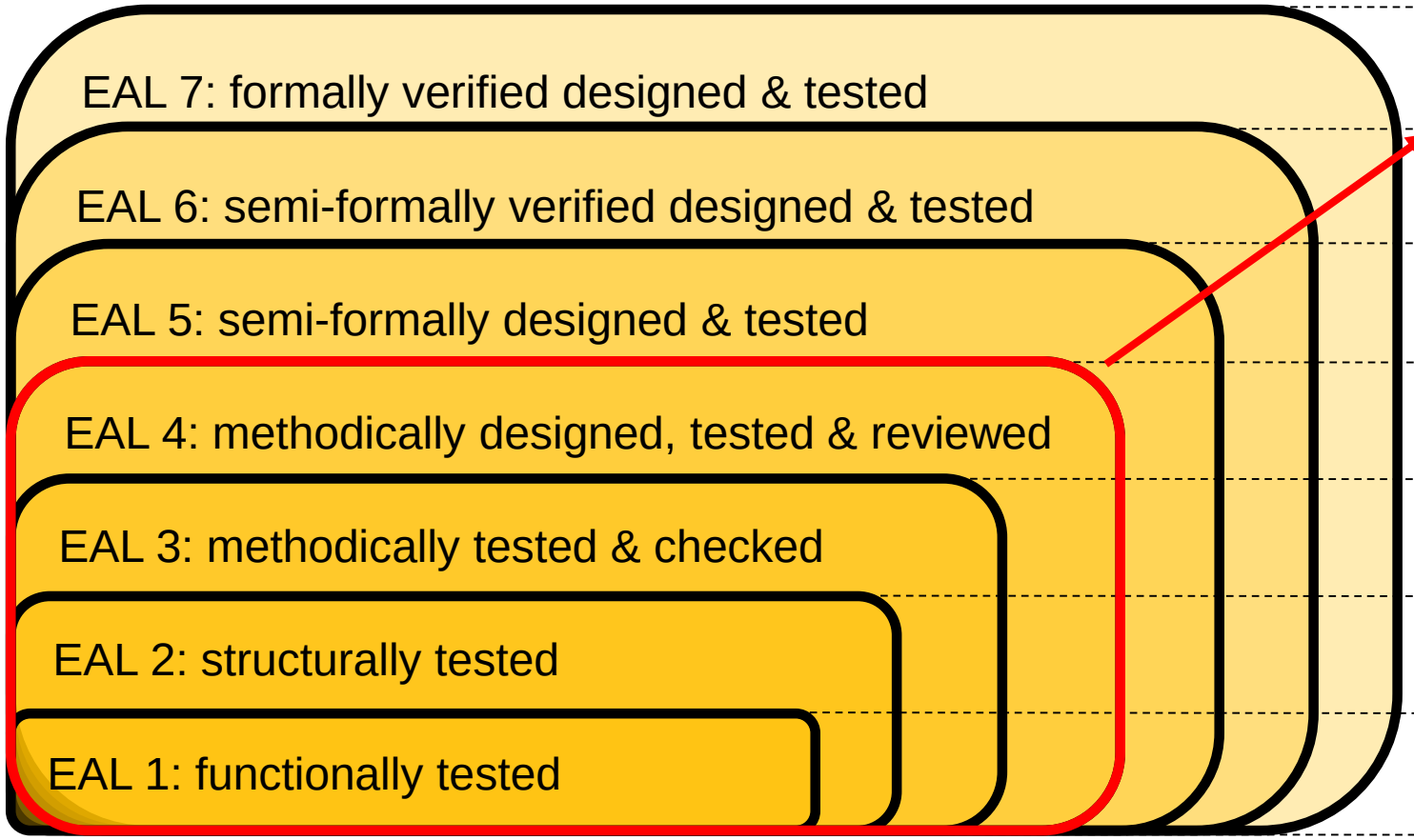
# General Workflow



# Security Evaluation acc. to Common Criteria (CC) – Smart Cards



# Evaluation Assurance Levels (EALs)



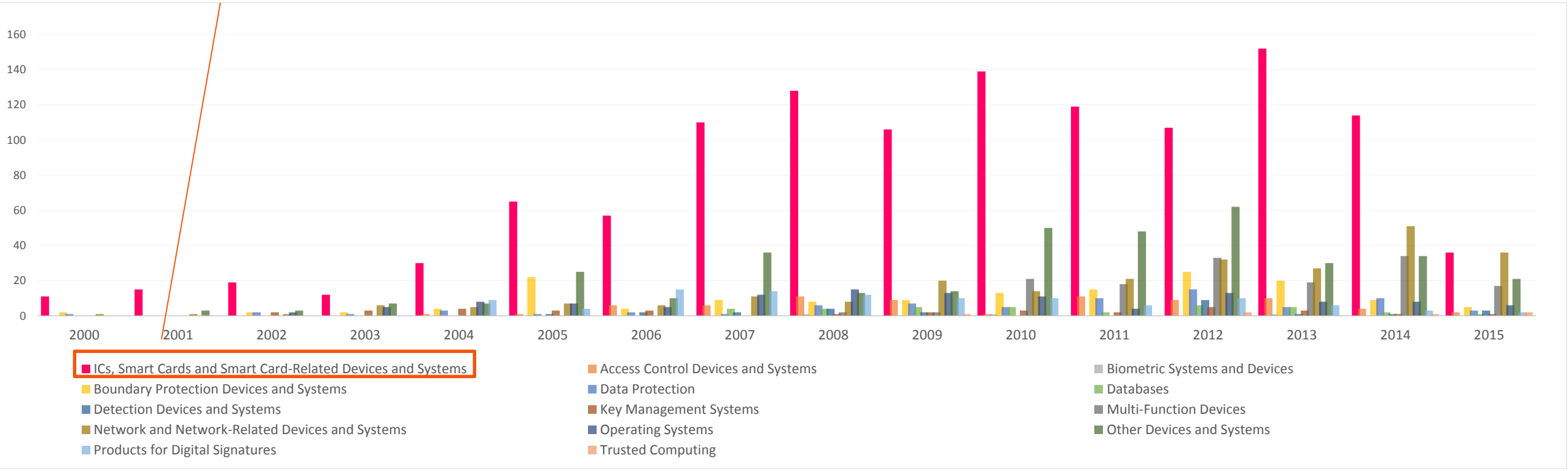

| Vulnerability Assessment Level | |
|---|---|
| AVA_VAN.5 | High |
| AVA_VAN.5 | High |
| AVA_VAN.4 | Moderate |
| AVA_VAN.3 | Enhanced-Basic |
| AVA_VAN.2 | Basic |
| AVA_VAN.2 | Basic |
| AVA_VAN.1 | No |

Augmentation (noted as “+” e.g. EAL5+): at least one component from a higher level has been taken

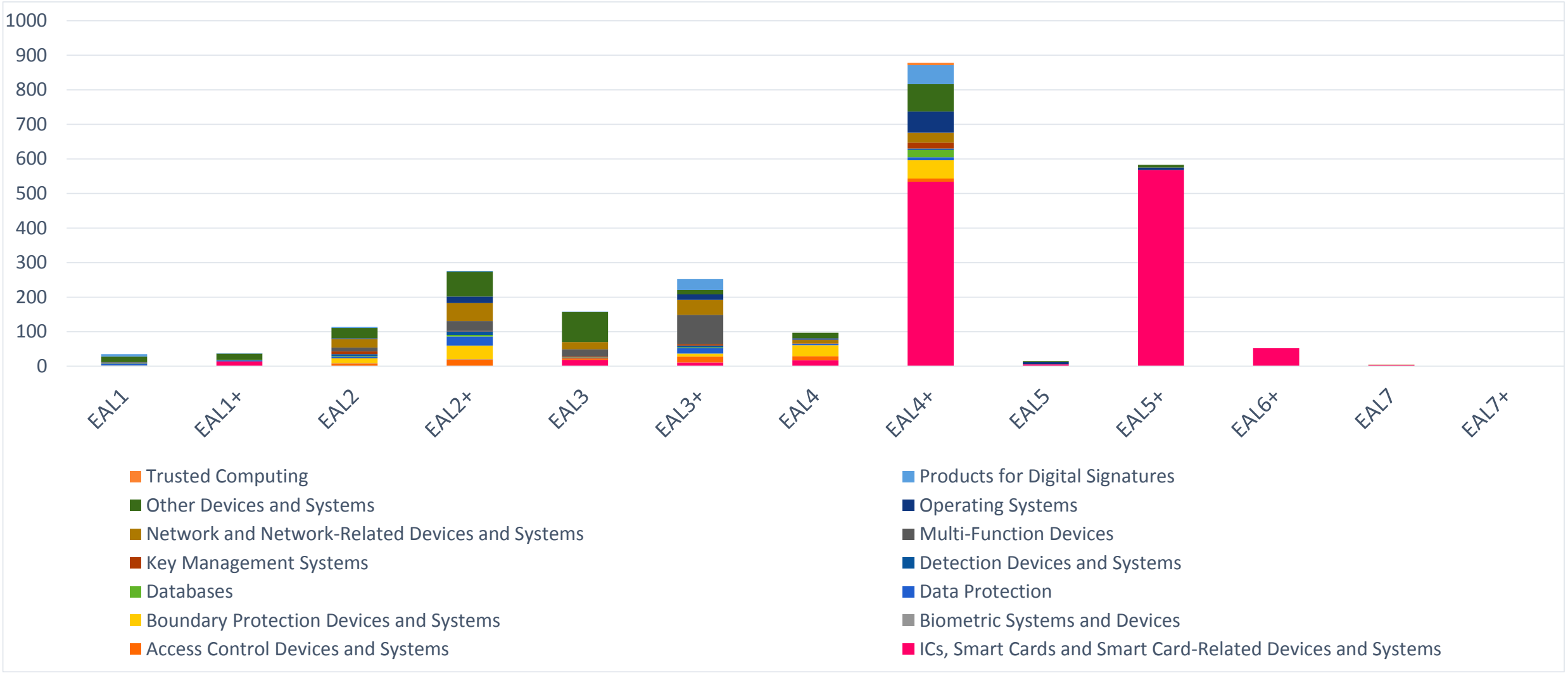
For Smart Cards according to protection profile PP0084 the assurance level MUST BE at least EAL4+, where + is always AVA_VAN.5

# Number of CC Certificates in the Different Domains

ICs, Smart Cards and Smart Card-Related Devices and Systems



# Number of Certificates per Domain and EAL



# NXP (Composite) Certifications: Payment vs. eID

Payment

eID

**VISA / MasterCard / AMEX**

**EMVCo Platform Approval**

**EMVCo HW Approval**

| | |
|---|---|
| Paym. Applet | ID Applet |
| Java Card OS | |
| Cryptographic Library | |
| SmartMX2 HW | |

**PassPort App CC Certification EAL5+**

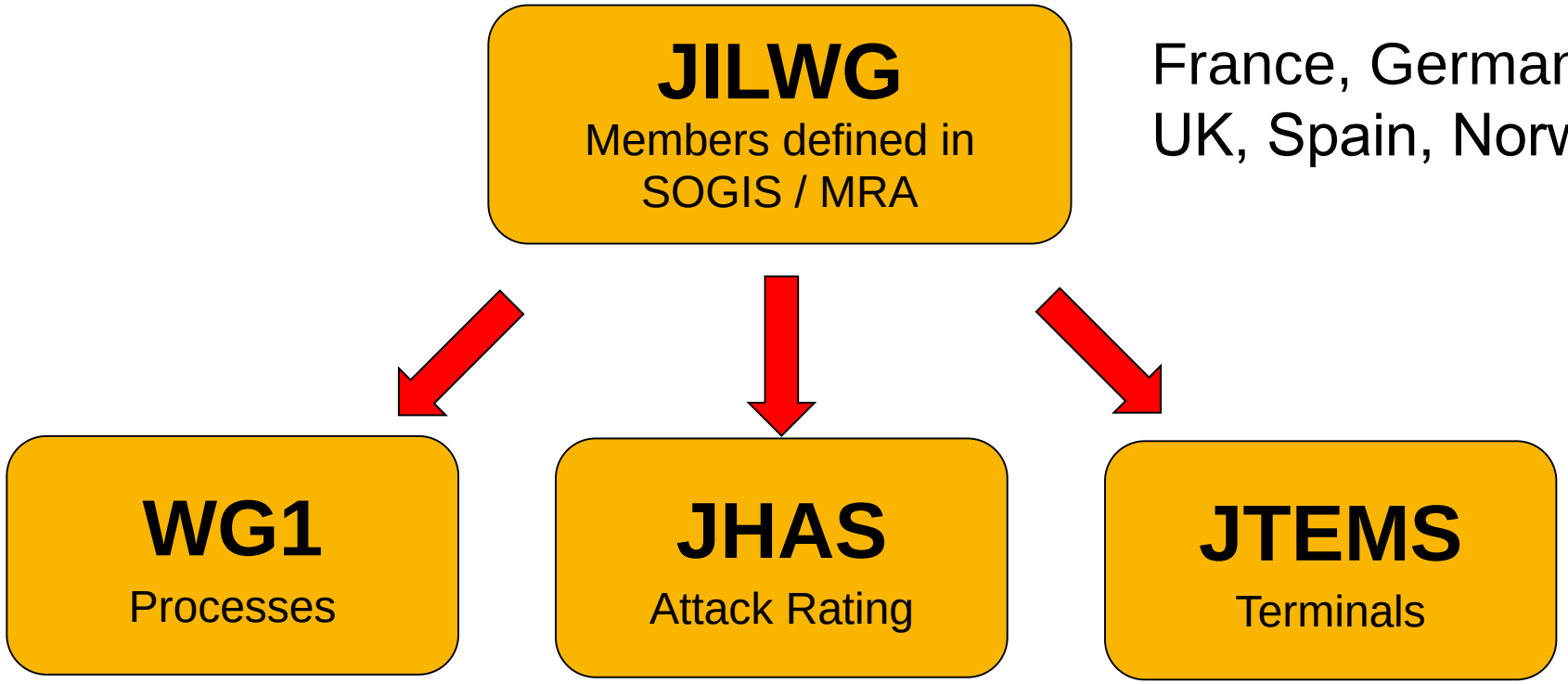
**JCOP CC Certification EAL6+**

**CryptoLib CC Certification EAL6+**

**HW CC EAL6+**

# JHAS ATTACK RATING

# Working Groups in the Smart Card Domain




France, Germany, Netherlands, UK, Spain, Norway,…

# Major Attack Classes

- Physical Attacks (e.g. Reverse Engineering)
- Overcoming Sensors and Filters (e.g. Disconnection)
- Perturbation Attacks (e.g. Skipping instruction)
- Side-channel Attacks (e.g. extract key bits from power profile)
- ...
- Exploitation of Test Features (e.g. set TOE back to test mode)
- Attacks on RNG (e.g. influence quality)
- Software Attacks (e.g. replay attacks)

# Attack Phases

- **Identification Phase**:
  Perform the attack once to demonstrate its feasibility and / or achieve a one-time benefit (learning phase)

- **Exploitation Phase**:
  Perform the attack multiple times for commercial exploitation

- **Information Flow between these Phases**:
  One of the outcomes of the Identification Phase is a virtual script that tells the attacker of the Exploitation Phase how to perform the attack

# Rating Tables for Smart Cards

| Range of values CC 3.1 | TOE resistant to attackers with attack potential of: |
|---|---|
| 0-15 | No rating |
| 16-20 | Basic |
| 21-24 | Enhanced-Basic |
| 25-30 | Moderate |
| 31 and above | High |

- For Smart Cards we need to achieve 31 points for
  - CC AVA_VAN.5 (part of EAL 4+, 5, 5+, 6, 6+, 7)
  - EMVCo
  - MasterCard
  - VISA
  - AMEX, …
- for each and every attack path!

| Factors | Identification | Exploitation |
|---|---|---|
| **Elapsed time** | | |
| < one hour | 0 | 0 |
| < one day | 1 | 3 |
| < one week | 2 | 4 |
| < one month | 3 | 6 |
| > one month | 5 | 8 |
| Not practical | * | * |
| **Expertise** | | |
| Layman | 0 | 0 |
| Proficient | 2 | 2 |
| Expert | 5 | 4 |
| Multiple Expert | 7 | 6 |
| **Knowledge of the TOE** | | |
| Public | 0 | 0 |
| Restricted | 2 | 2 |
| Sensitive | 4 | 3 |
| Critical | 6 | 5 |
| Very critical hardware design | 9 | NA |
| **Access to TOE** | | |
| < 10 samples | 0 | 0 |
| < 100 samples | 2 | 4 |
| > 100 samples | 3 | 6 |
| Not practical | * | * |
| **Equipment** | | |
| None | 0 | 0 |
| Standard | 1 | 2 |
| Specialized | 3 | 4 |
| Bespoke | 5 | 6 |
| Multiple Bespoke | 7 | 8 |
| **Open samples** | | |
| Public | 0 | NA |
| Restricted | 2 | NA |
| Sensitive | 4 | NA |
| Critical | 6 | NA |

# SUMMARY

# Summary

- A lot of **solutions exist to build secure systems** – at least on paper

- **Implementing them securely** – especially on embedded systems – is extremely **challenging (facing physical attacks)**

- To **convince customers** that a system is secure and implemented securely requires **security evaluation & certification**

- The **level of** implementation **security** needs to be finally **rated** (e.g. JHAS attack rating) which expresses the **costs of attacks to break the product**

SECURE CONNECTIONS
FOR A SMARTER WORLD

# ATTRIBUTION STATEMENT

NXP, the NXP logo, NXP SECURE CONNECTIONS FOR A SMARTER WORLD, CoolFlux, EMBRACE, GREENCHIP, HITAG, I2C BUS, ICODE, JCOP, LIFE VIBES, MIFARE, MIFARE Classic, MIFARE DESFire, MIFARE Plus, MIFARE FleX, MANTIS, MIFARE ULTRALIGHT, MIFARE4MOBILE, MIGLO, NTAG, ROADLINK, SMARTLX, SMARTMX, STARPLUG, TOPFET, TrenchMOS, UCODE, Freescale, the Freescale logo, AltiVec, C 5, CodeTEST, CodeWarrior, ColdFire, ColdFire+, C Ware, the Energy Efficient Solutions logo, Kinetis, Layerscape, MagniV, mobileGT, PEG, PowerQUICC, Processor Expert, QorIQ, QorIQ Qonverge, Ready Play, SafeAssure, the SafeAssure logo, StarCore, Symphony, VortiQa, Vybrid, Airfast, BeeKit, BeeStack, CoreNet, Flexis, MXC, Platform in a Package, QUICC Engine, SMARTMOS, Tower, TurboLink, and UMEMS are trademarks of NXP B.V. All other product or service names are the property of their respective owners. ARM, AMBA, ARM Powered, Artisan, Cortex, Jazelle, Keil, SecurCore, Thumb, TrustZone, and µVision are registered trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. ARM7, ARM9, ARM11, big.LITTLE, CoreLink, CoreSight, DesignStart, Mali, mbed, NEON, POP, Sensinode, Socrates, ULINK and Versatile are trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org. © 2015–2016 NXP B.V.