Physical Attacks and Tamper Resistant Devices

Levente Buttyán

CrySyS Lab, BME

www.crysys.hu

Contents

- What are tamper resistant devices and where do we need them?
- Taxonomy of and examples for physical attacks
- FIPS 140 security levels
- Examples for tamper resistant devices
- API attacks

What are tamper resistent devices?

- tamper resistant computing devices are designed for carrying out operations in strict isolation from their environment
  - they **can store and process data** such that neither the data nor any execution state can be observed or modified by an attacker, **even with direct physical access** to the device
  - stored data typically include cryptographic keys that never leave (in unencrypted form) the physically secure environment provided by the device
  - such devices eliminate the need to physically protect an entire computing system (e.g., computer)

- examples:
  - smart cards
  - TPM chips
  - hardware security modules (HSM)

- typical features:
  - limited and well controlled interfaces
  - access control enforcing a security policy
  - passive/active tamper prevention/detection and response

Use case – Public Key Infrastructures

- private keys of users and CAs must be protected from disclosure
- the private keys of users are typically stored on smart cards
  - the smart card can control the usage of the private key
  - the smart card can run cryptographic algorithms such that the private key does not need to leave the protected environment
- CA private keys are held in tamper resistant HSMs (Hardware Security Modules) that help enforce stringent policies on key usage:
  - they can enforce dual control policies on the most valuable keys
  - they can help supervisors monitor the activities of large numbers of human operators efficiently
  - they can keep signed audit trails of activities to allow retrospective analysis of access operations

Use case – ATM security

- PIN derivation
  - customer PIN is derived from account number and a PIN derivation key
  - the PIN derivation key needs to be protected against unauthorized disclosure (including insiders such as bank personnel)

- PIN verification
  - customer PIN is stored in an encrypted form on the customer's card
  - verification at an ATM needs either the PIN decryption key or secure communication of the entered PIN from the ATM to the bank
  - in both cases, keys need to be protected against unauthorized disclosure (including insiders such as ATM maintenance personnel)

Use case – Automated fare collection

- electronic ticketing for public transportation
  - all transactions can be logged, collected, and analyzed
  - efficiency of the system can be increased by careful planning of schedules

- e-tickets may store value, in which case they must be protected from manipulation

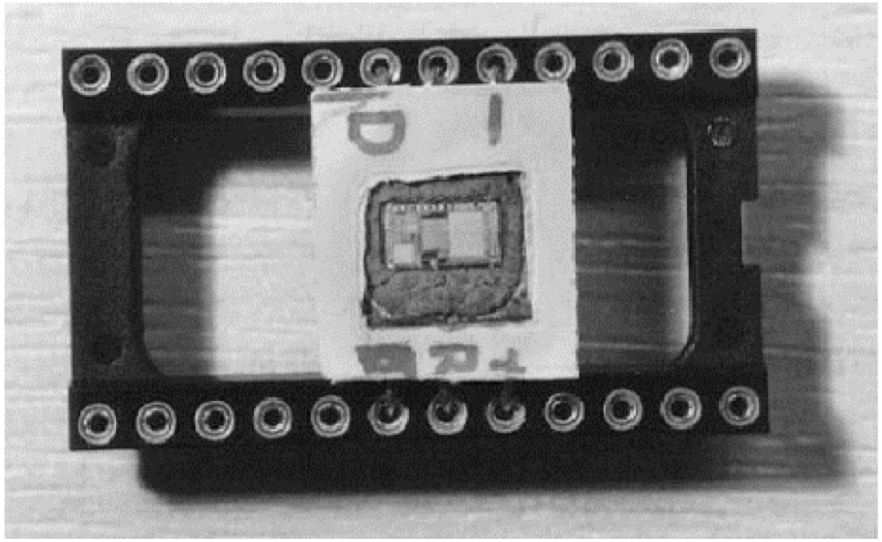
- e-tickets may need to be authenticated during the validation process
  - authentication is based on a key derived from the ticket ID and a master key (key diversification)
  - master key must be stored in off-line ticket validating equipment (e.g., on buses, trams, etc)
  - master key needs to be protected from disclosure when equipment is stolen

Taxonomy of attacks (on tamper resistant chips)

- **invasive attacks**
  - direct access to the internal components of the chip
  - often results in permanent damage
- **semi-invasive attacks**
  - access to the chip surface without damaging its passivation layer
  - the chip remains operational
  - electrical contacts through authorized interfaces only, but…
  - influence on internal operations by using some physical phenomenon
- **non-invasive attacks**
  - no physical penetration to the chip
  - **local** non-invasive attacks:
    - » observation or manipulation of the interaction between the device and its physical environment (timing, power consumption, clock frequency)
  - **remote** non-invasive attacks:
    - » observing or manipulating only the normal input and output of the device (API attacks)

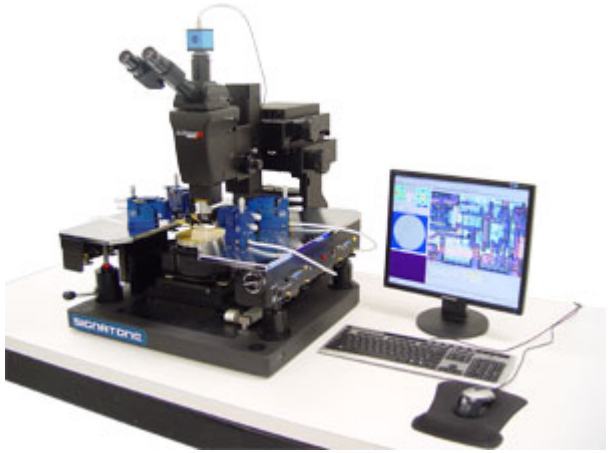
Invasive attacks – first step

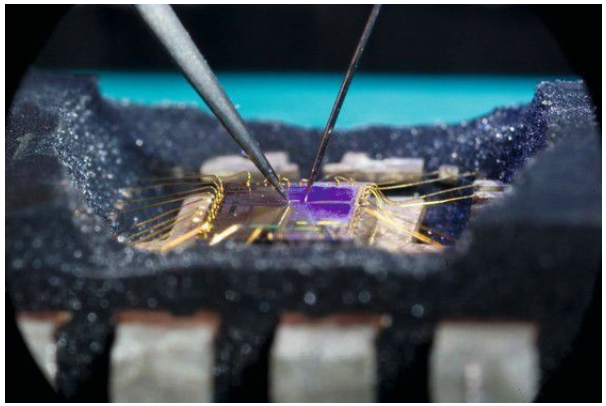
- gain access to the internal components of a device by removing its cover or drilling holes
- in case of chips, removing the cover typically requires chemical processes (e.g., nitric acid dissolves epoxy without damaging silicon)



Invasive attacks – second step

- probing chip internals directly
  - second-hand semiconductor test equipment such as manual probing stations are available for renting
  - a typical probing station consists of
    - » a **microscope** with an objective working distance of several millimeters mounted on a **low-vibration platform**
    - » **micromanipulators** to place probes (microprobing needles) on to the device
    - » a **laser**, with which small holes can be drilled in the chip's passivation layer (holes allow electrical contact by the probes, and indeed stabilise them in position)
  - with such equipment one can probe the device's internal bus system, so that both program and data can be read out

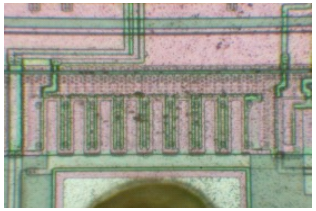




Defending against invasive attacks

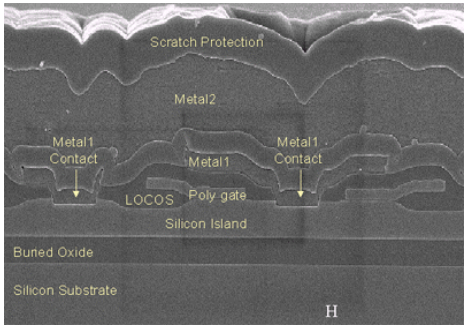
- sensor mesh implemented in the top metal layer, consisting of a serpentine pattern of sensor, ground and power lines
  - if the sensor line is broken, or shorted to ground or power, the device deletes sensitive data

- making it more difficult to visually analyze the chip surface
  - earlier the structure of a microcontroller could be easily observed and reverse engineered under a microscope
  - although buried under the top metal layer, the second metal layer and polysilicon layer can still be seen, because each subsequent layer in the fabrication process follows the shape of the previous layer
  - today, each layer but the last one is planarised using chemical-mechanical polishing before applying the next layer
  - the only way to reveal the structure of the deeper layers is by removing the top metal layers either mechanically or chemically



PIC16F877

PIC16F877A



Semi-invasive attacks

- attacks that involve access to the chip surface, but which do not require penetration of the passivation layer or direct electrical contact to the chip internals

- early example: UV light to reset the protection bit on microcontrollers (memory contents could be read out)

- recently: optical probing techniques to inject faults into digital circuits
  - illuminating a target transistor causes it to conduct, thereby inducing a transient fault
  - possible to set or reset any individual bit of SRAM in a microcontroller
  - can be carried out with simple, low-cost equipment (e.g., with photographer's flash gun)

- better results can be obtained using laser probing equipment
  - in addition to setting RAM contents to desired values, it can be adapted to read out other memory technologies, such as Flash and EEPROM, and to interfere with control logic directly

Defending against semi-invasive attacks

- detection of active attacks and performing some suitable alarm function, such as erasing key material
- opaque top-layer that shields the chip from external light
  - the attacker may now have to go through the rear of the chip, which will typically involve ion-etching equipment to thin the device and an infra-red laser to penetrate the silicon substrate

Local non-invasive attacks

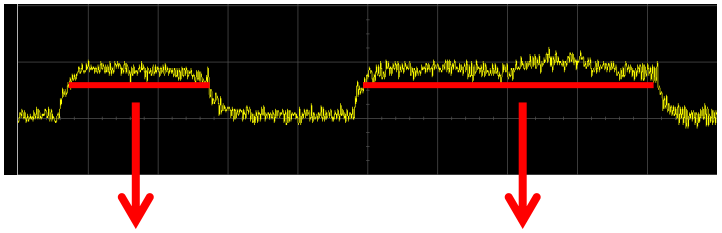
- side-channel attacks
  - careful observation of the interaction of the device with its environment during critical operations may reveal some amount of information about the sensitive data stored in the card
  - timing attacks and **power analysis**
- fault injection attacks
  - unusual operating conditions may have undocumented effects
    - » e.g., unusual temperatures or voltages can affect EEPROM write operations
  - power and clock glitches may affect the execution of individual instructions
    - » e.g., doubling the clock frequency for a few microseconds would cause some, but not all, instructions to fail
    - » it can be possible to modify the device's control flow by stepping over the branch instruction following a failed password check

Simple Power Analysis (SPA)

- the power consumption trace of the device reflects the aggregate activity of its individual elements
- the power consumption may be data (secret key) dependent
- SPA examines features that are directly visible in a trace (maybe in a single trace!)
- example: observing the power trace of RSA modular exponentiation with the ”square and multiply” method

```
inputs: ciphertext Y
        private exp d = [d_k d_k-1 ... d_1]
X = 1
for i = k to 1 do
    X = X² mod n
    if d_i == 1 then X = X*Y mod n
end for
output X
```

power consumption:



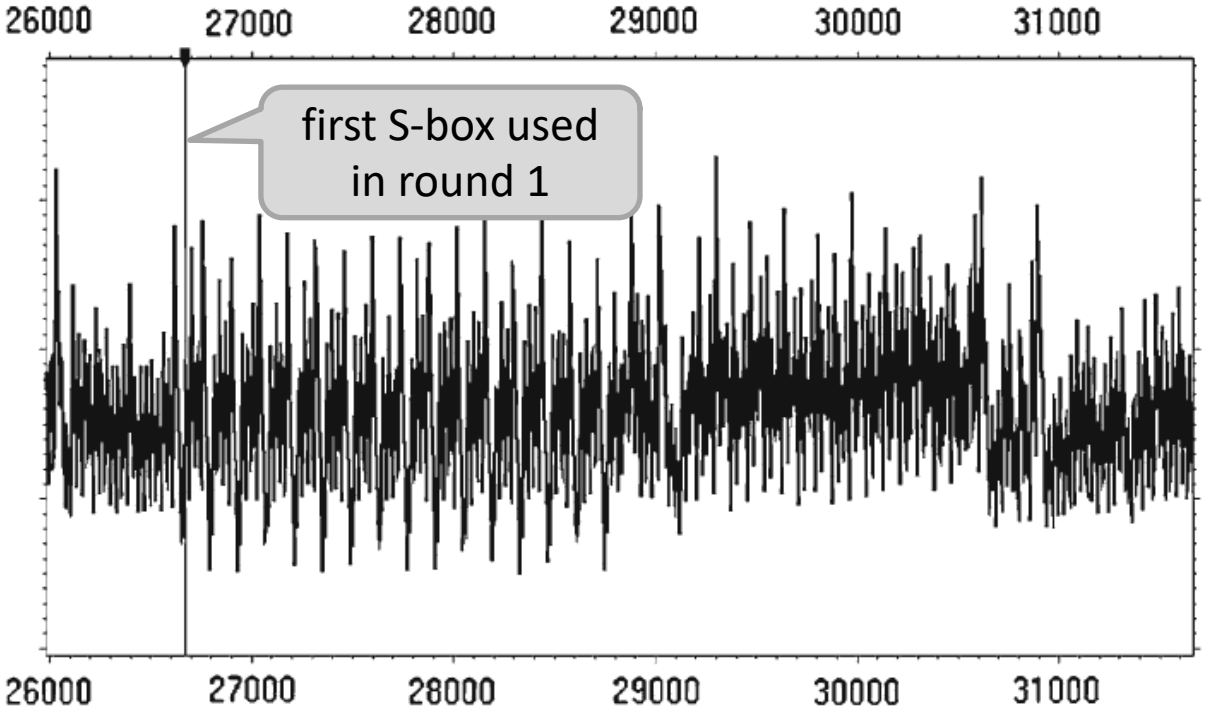
$X = X^2 \bmod n$
$\rightarrow d_i = 0$

$X = X^2 \bmod n$
$X = X*Y \bmod n$
$\rightarrow d_i = 1$

Differential Power Analysis (DPA)

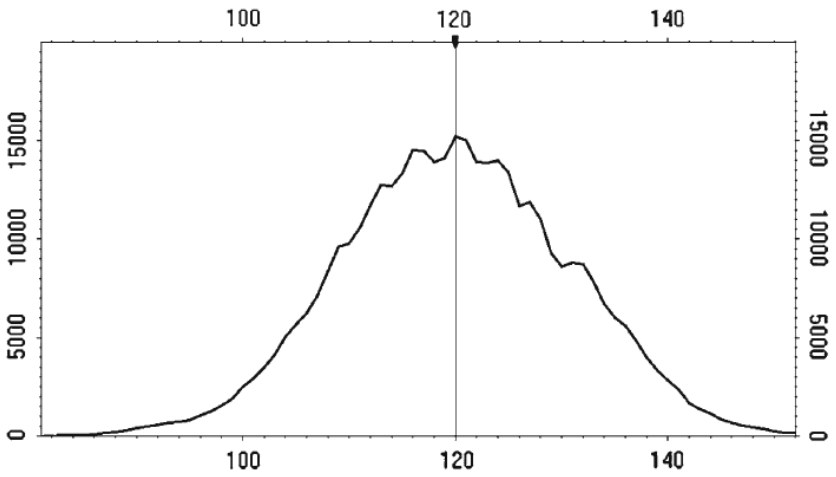
- bits of the key cannot typically be read from a single power trace
  - dependence on key bits is not so direct
  - measurements are noisy
- DPA is a statistical method that uses sets of measurements to eliminate noise and to identify data-dependent correlations
  - obtain a set of power traces from a chip while it performs cryptographic computation on different inputs
  - guess the value of some bits of the key (e.g., 1 byte of the key)
  - sort the power traces into two piles, depending on whether the guessed key bits combined with the known inputs would have activated some computation (circuit inside the chip) or not
  - check the guess by verifying if the two piles are statistically different
    - » if the guess was wrong, there shouldn't be any significant statistical difference between the two piles
    - » if the guess was correct, traces in the two piles are indeed different (in one of the piles, we have traces where the given circuit was active, in the other, we have traces where the circuit was not active)

DPA example

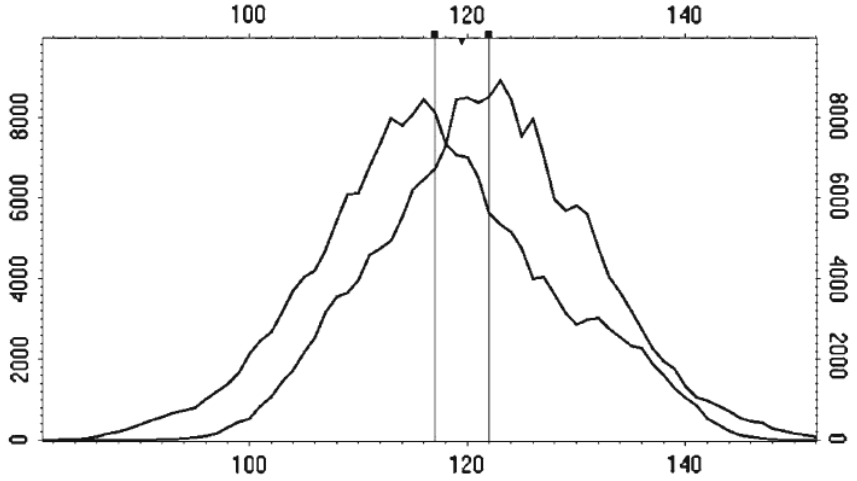


Power trace of AES-128 encryption on a smart card

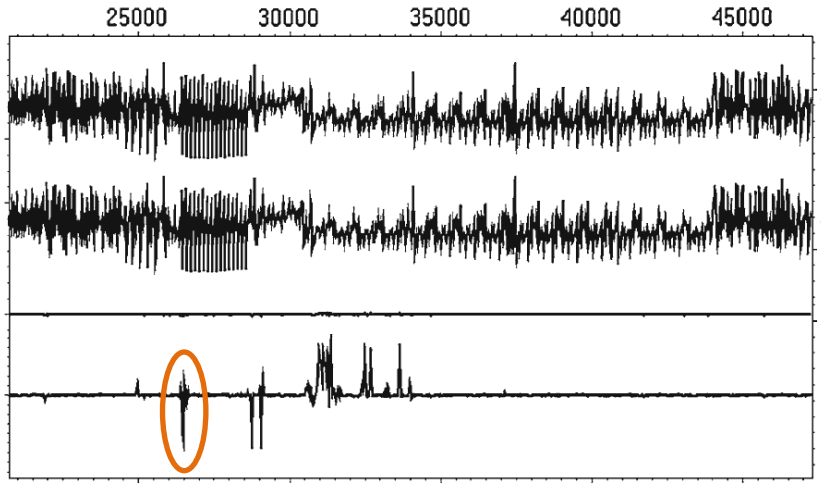
DPA example



Distribution of power consumption at first S-box output computation (computed over thousands of traces)



Distributions of power consumption for traces with the LSB of the output of the first S-box being 1 (left) and 0 (right)



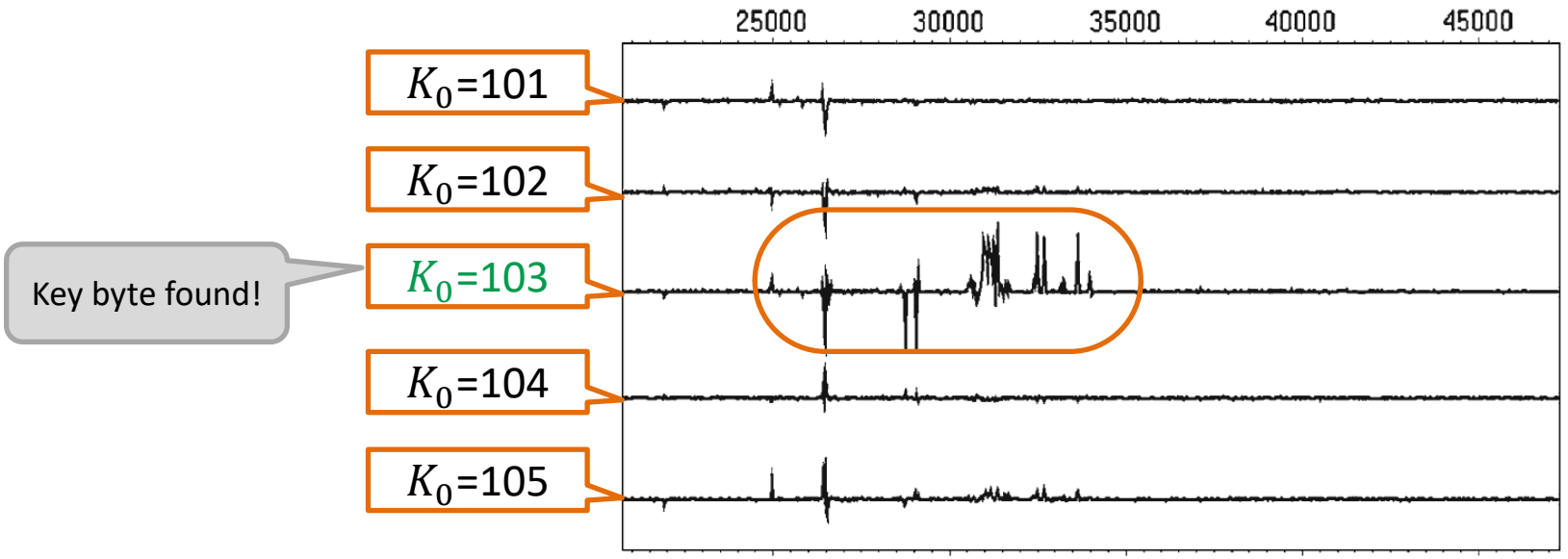
average of traces where LSB is 1

average of traces where LSB is 0

difference (x15)

DPA example

- output of the first S-box depends only on one byte of the key
- for a guess of this key byte and a known input, we can determine the LSB of the output of the S-box
- we sort the traces (obtained for different inputs) into two piles based on the computed LSB
- we compute the difference of the average traces of the piles and check for spikes…



Defending against side channel attacks

- eliminate conditional execution of steps
- make timing and power consumption of different execution branches identical
  - can degrade performance if all branches are made similar to the slowest branch
- use constant time elementary operations
  - no dependence on input values (such as in case of table look-ups)
- make leaked information unrelated to secret data
  - typically based on some form of randomization of the inputs
  - input is randomized (blinded) → operation performed → randomization removed from result (unblinding)

FIPS 140 security levels

benchmark standard that specifies the security requirements for cryptographic modules

- level 1
  - **no physical security** mechanisms are required in the module
  - example: crypto library software on a PC
- level 2
  - needs **tamper evident** coating or seals
  - example: cheap microcontrollers, simple smart cards
- level 3
  - needs **tamper resistant** physical security
  - example: TPM chips, modern smart cards
- level 4
  - active **tamper detection and response** (immediately erasing all secret data)
  - protection against compromise due to environmental conditions or fluctuations outside of the normal operating ranges (e.g., voltage, temperature, …)
  - example: hardware security modules (HSM), crypto co-processors (IBM 4758)

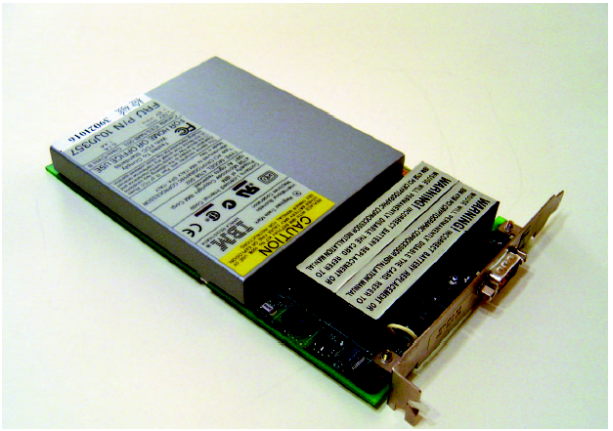
Classes of tamper resistant devices

- high-end devices:
  - e.g., IBM 4758 coprocessor
  - powerful crypto engine surrounded by a tamper-sensing mesh
  - **active detection of tampering attampts** → device erases its key material and renders itself inoperable

- low-end devices:
  - e.g., cheap microcontrollers
  - typically capable only for symmetric key crypto
  - their physical and logical protection mechanisms are **not really designed to withstand skilled and determined attacks**

- mid-range:
  - e.g., smart cards and TPM chips
  - single-chip products **hardened against physical attacks**
  - hardware support for crypto

The IBM 4758 crypto coprocessor



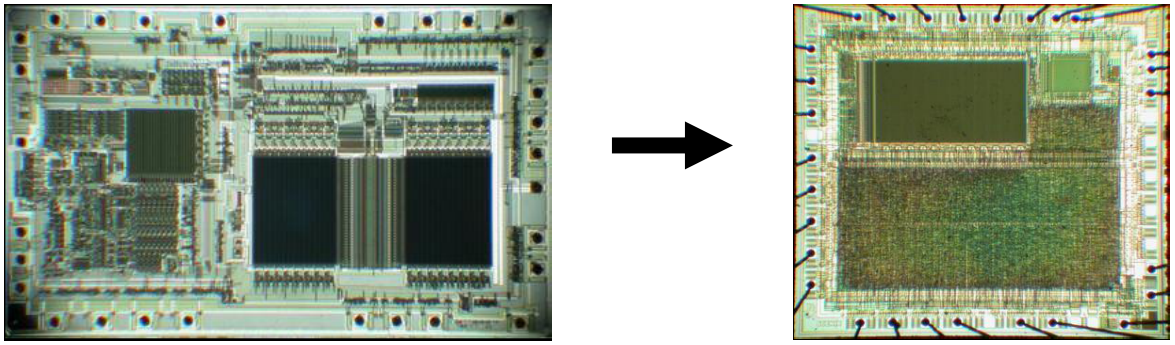
- programmable PCI board with tamper resistant packaging
- main features:
  - pipelined encryption engine
  - pipelined hash engine
  - modular math hardware to support RSA and DSA
  - hardware noise source to seed random number generation
  - pseudo-random number generator
  - support for RSA key pair generation, encryption, and decryption
  - support for key management, key diversification, PIN generation
  - secure clock-calendar
  - support for PKCS#11 and IBM Common Cryptographic Architecture (CCA)
  - battery backed RAM (BBRAM) to store secrets persistently
  - steel house with tamper detecting sensors and circuitry to erase the sensitive memory

IBM 4758: Withstanding physical attacks

- the board is wrapped in a grid of conductors, which is monitored by a circuit that can detect changes in the properties of these conductors
  - conductors are non-metallic and resemble the material in which they are embedded
  - the grid is arranged in several layers
- entire package is enclosed in a grounded shield to reduce detectable electromagnetic emanations
- additional sensors:
  - temperature
  - humidity
  - pressure
  - ionizing radiation
  - changes in supply voltage and clock frequency
- reaction to tamper: erase BBRAM and reset the whole device
- physical security is certified at FIPS 140-2 level 4

Physical security measures of smart cards

- top-layer conductor meshes
- randomized ASIC-like logic design (glue logic)
- internal bus hardware encryption to make data analysis and access to internal memory more difficult
- light sensors to prevent an opened chip from functioning
- voltage sensors to protect against under- and over-voltages used in power glitch attacks
- clock frequency sensors to prevent attackers slowing down the clock frequency for static analysis and also from raising it for clock-glitch attacks



API attacks

- non-invasive attacks that can be carried out even without physical access to the device
- the API of a tamper resistant module is a software layer through which the module's functions are exposed to the external world
- attacking the API means exploiting design weaknesses of the API for extracting secrets from the device (or just increasing the efficiency of cryptanalytical attacks)

<u>simple example:</u>

- API:

  encrypt ($E_{MK}(K)$, msg) → returns E_K(msg)

  decrypt ($E_{MK}(K)$, E_K(msg)) → returns msg

  key_export ($E_{MK}(K)$, E_{MK}(KEK)) → returns $E_{KEK}(K)$

- a typing attack:

  key_export ($E_{MK}(K)$, E_{MK}(KEK)) → returns $E_{KEK}(K)$

  decrypt (E_{MK}(KEK), $E_{KEK}(K)$) → returns K

PIN management on the IBM 4758 coprocessor

- the IBM 4758 uses IBM's Common Cryptographic Architecture API (CCA API)
- the CCA API supports PIN management functions (called financial services support):
  - PIN generation
  - PIN encryption
  - transformations between encrypted versions of the PIN (translation from one key to another)
  - encrypted PIN verification
  - changing the PIN
  - CVV generation and verification
  - …

CCA API – PIN generation

- inputs:
  - PAN (Personal Account Number)
  - decimalization table (mapping of hex digits to decimal digits)
  - PIN generation key ID / token
- steps:
  - encrypt PAN with the PIN generation key
  - decimalize the result with the decimalization table supplied
  - select and return the desired amount of digits --» PIN
- secure output (e.g., to secure printer):
  - PIN

- example:
  - PAN = 87654321 87654321
  - ciphertext = a0b1c2d3e4f5a6b7 (hex)
  - decimalization table = 0123456789012345
  - decimalized ciphertext = 0011223344550617
  - four digit PIN = 0011

CCA API – Encrypted PIN verification

- input:
  - PAN
  - decimalization table
  - encrypted PIN block
  - PIN encryption key ID / token
  - PIN generation key ID / token
- steps:
  - compute PIN from PAN, PIN generation key, and dec table --» PIN
  - decrypt encrypted PIN with PIN encryption key --» PIN’
  - compare PIN to PIN’
- output:
  - accept / reject (/ PIN formatting error)

CCA API – PIN offset generation (PIN change)

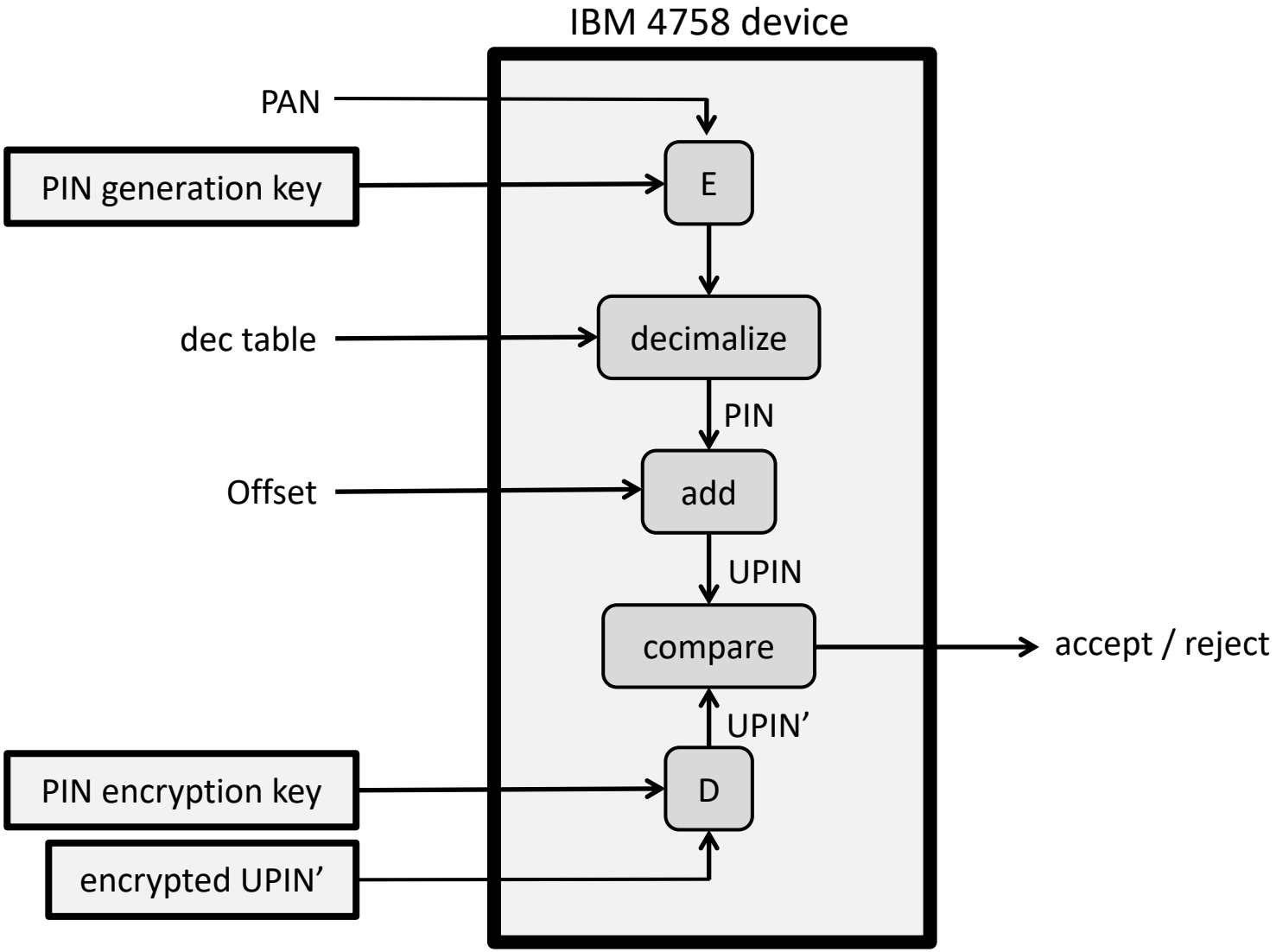
- input:
  - PAN
  - decimalization table
  - PIN generation key ID / token
  - user selected PIN (UPIN)
- steps:
  - generate PIN from PAN, PIN generation key, and decimalization table --» PIN
  - digit subtraction: UPIN – PIN (mod 10) --» OFFSET
- output:
  - OFFSET (stored on the user's card)

- example:
  - PAN = 87654321 87654321
  - ciphertext = a0b1c2d3e4f5a6b7 (hex)
  - decimalization table = 0123456789012345
  - decimalized ciphertext = 0011223344550617
  - four digit initial PIN = 0011
  - user selected UPIN = 1234
  - OFFSET = 1223

CCA API – Encrypted PIN verification with offset

- input:
  - PAN
  - decimalization table
  - encrypted PIN block (contains user typed UPIN')
  - Offset
  - PIN encrypting key ID / token
  - PIN generation key ID / token
- steps:
  - compute initial PIN from PAN, dec table, and PIN generation key --» PIN
  - compute UPIN from PIN and Offset --» UPIN
  - decrypt encrypted PIN block with PIN encryption key --» UPIN'
  - compare UPIN to UPIN'
- output:
  - accept / reject / error

Encrypted PIN verification with offset – illustrated



CCA API – Decimalization attack

- example setup:

```
PAN =          1122334455667788
enc PAN =      E481FC5658391418
dec table =    0123456789012345
PIN =          4481
UPIN =         6598
Offset =       2117
```

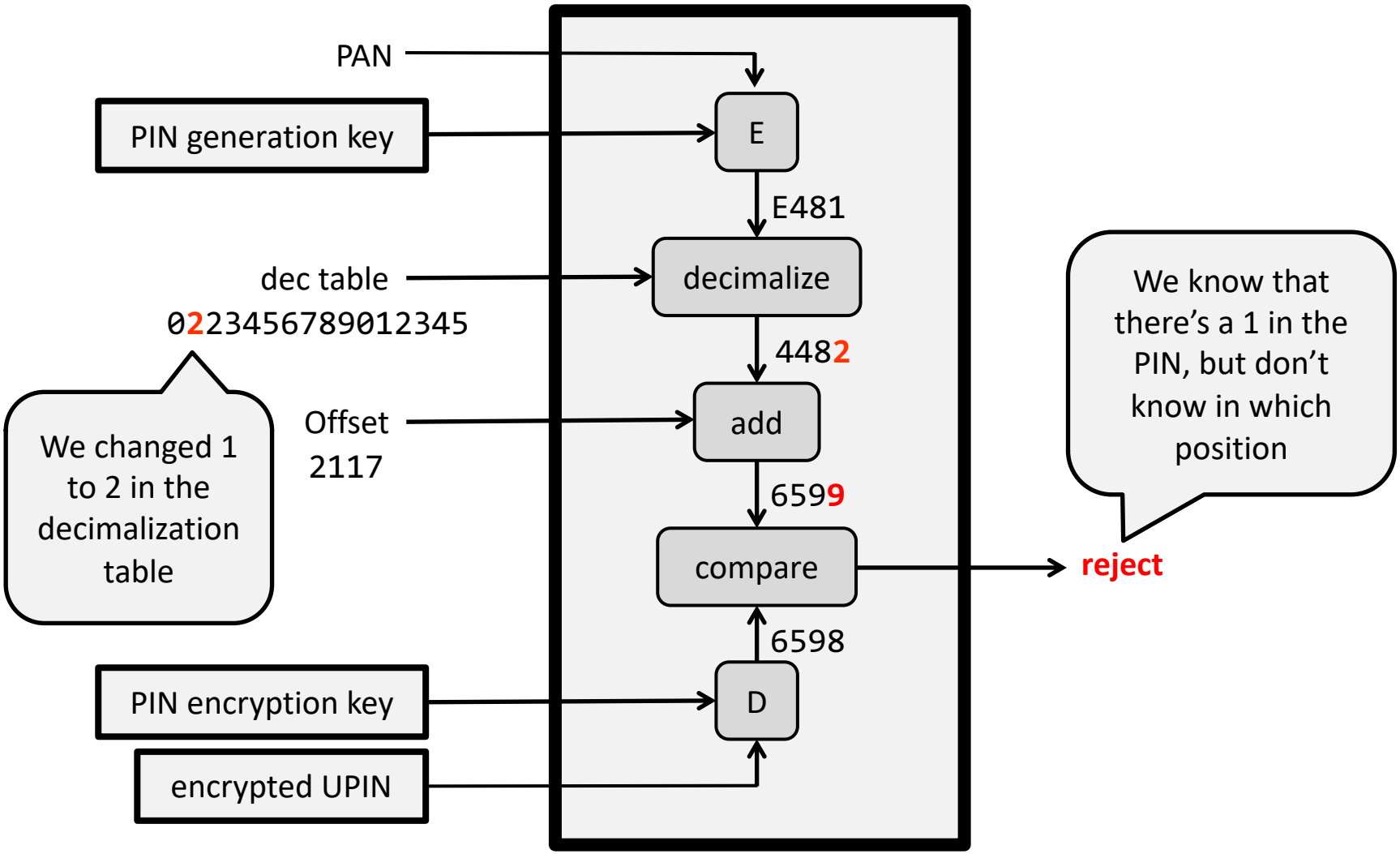
- assume the attacker can call “encrypted PIN verification with offset” and he can manipulate the input fields (has physical access to the device)
- what if we change the dec table to **1**123456789012345 ?
  - since E481 does not contain a 0, we get the same PIN = 4481, and Offset = 2117 will pass --» the device returns “accept”

CCA API – Decimalization attack

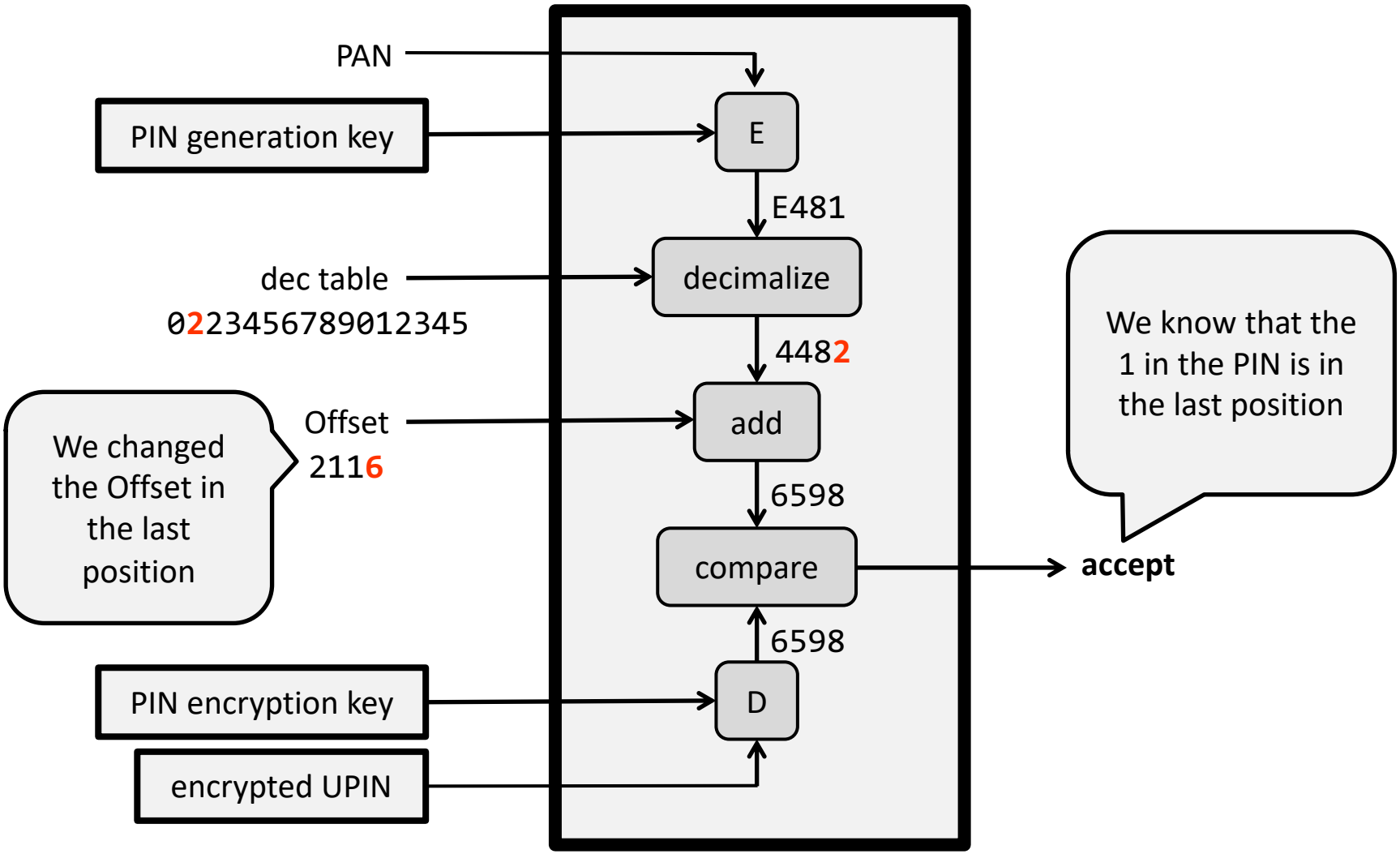
- now let's change the dec table to 0223456789012345 !
  - the encrypted PAN is decimalized into PIN = 4482
  - Offset = 2117 does not pass (4482 + 2117 = 6599)
  - we know that there's a 1 in the initial PIN !
  - we have to find out in which position
  - for this, we modify the Offset until it passes again
  - in this case, Offset = 2116 will pass (4482 + 2116 = 6598)
  - we know that the last digit of the initial PIN is 1 !

- and so on …
- when PIN is obtained, we compute UPIN as PIN + Offset

Decimalization attack – illustrated



Decimalization attack – illustrated



Lessons learned

- No matter how secure the device is physically if it leaks secrets due to API attacks
- API attacks can be very subtle and difficult to discover (the API may potentially have hundreds of functions)
- Most tamper resistant cryptoprocessors in the past were vulnerable to some form of API attacks
- Careful design and analysis of the API is indeed very important with respect to overall security of the device

Defending against API attacks

- the problem of API analysis seems to be very similar to that of analyzing authentication and key exchange protocols
  - the attacker interacts with the device using a well defined set of "messages"
  - the goal is to obtain some secret or bring the device in a "bad" state
- formal analysis techniques developed for key exchange protocols may be amenable to the analysis of crypto APIs

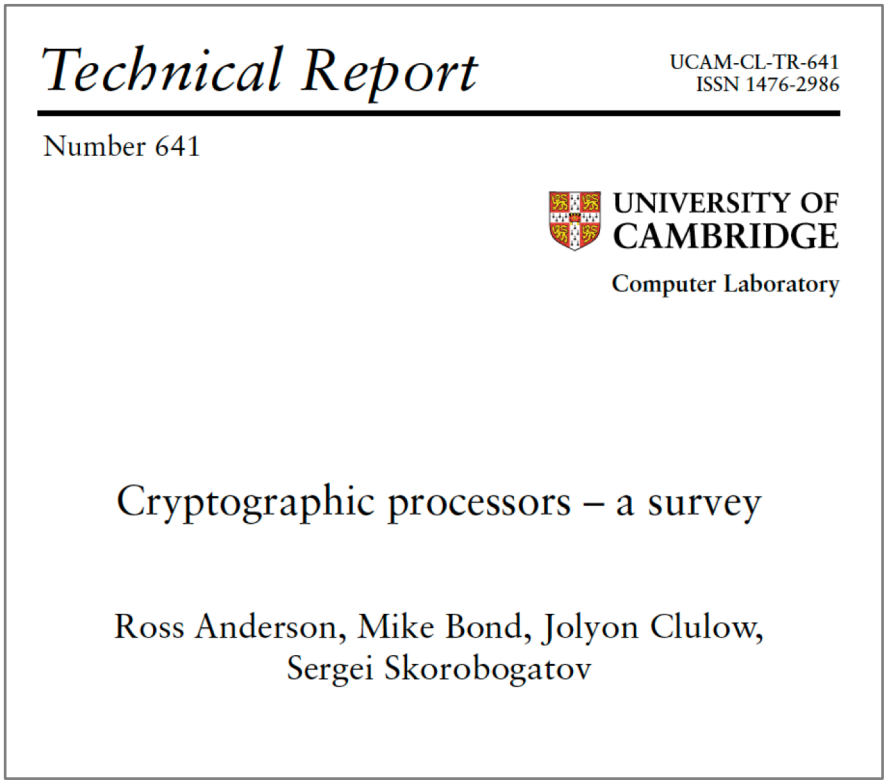
Security API analysis with the spi-calculus

Levente Buttyán *buttyan@crysys.hu*
Ta Vinh Thong *thong@crysys.hu*

Laboratory of Cryptography and Systems Security
Department of Telecommunications
Budapest University of Technology and Economics

Abstract: *API level vulnerabilities of hardware security modules represent a serious threat, thus, discovering and patching security holes in APIs are important. In this paper, we argue and illustrate that the application of formal verification methods is a promising approach for API analysis. In particular, we propose an API verification method based on process algebra. The proposed method seems to be extremely well-suited for API analysis as it allows for the straightforward modelling of the API, the precise definition of the security requirements, and the rigorous verification of the security properties offered by the API.*

Recommended readings



Technical Report

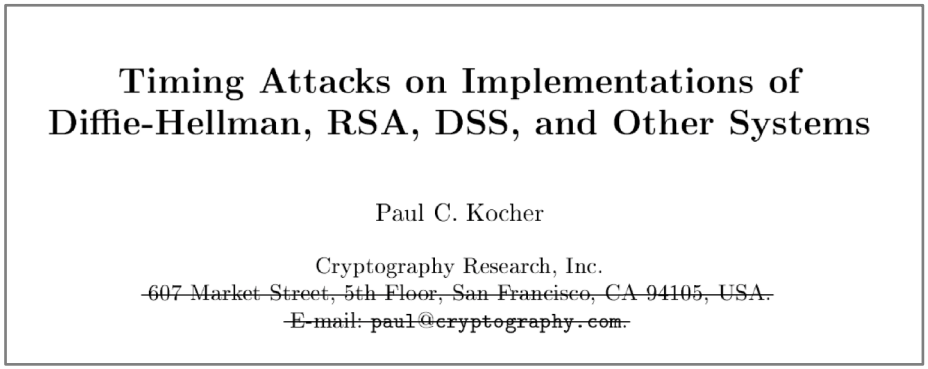
UCAM-CL-TR-641
ISSN 1476-2986

Number 641

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

Cryptographic processors – a survey

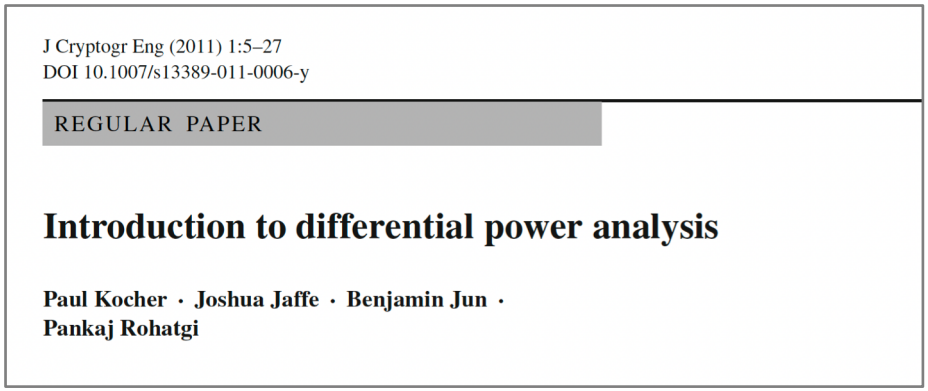
Ross Anderson, Mike Bond, Jolyon Clulow, Sergei Skorobogatov



Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems

Paul C. Kocher

Cryptography Research, Inc.
607 Market Street, 5th Floor, San Francisco, CA 94105, USA.
E-mail: paul@cryptography.com.



J Cryptogr Eng (2011) 1:5–27
DOI 10.1007/s13389-011-0006-y

REGULAR PAPER

Introduction to differential power analysis

Paul Kocher · Joshua Jaffe · Benjamin Jun · Pankaj Rohatgi

Control questions

- What do we mean by a tamper resistant device?
- Give some examples for application areas of tamper resistant devices!
- How can attacks against tamper resistant devices be classified?
- For each attack class, explain
  - how do those attacks work?
  - how can those attacks be prevented or detected?
- What levels of physical protection are defined in the FIPS 140 standard?
- Give some examples for high-end, mid-range, and low-end tamper resistant devices and their capabilities and physical protection features!
- What are API attacks and how can they be prevented?