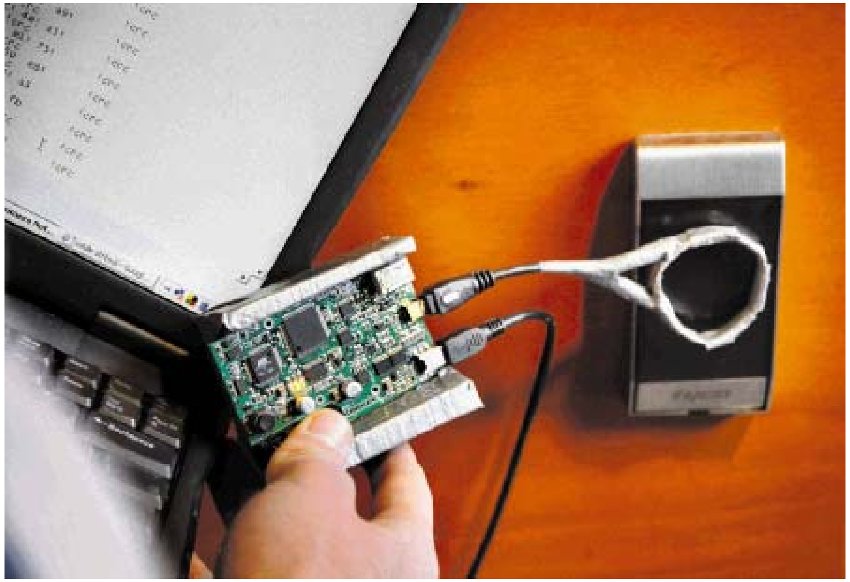
Hacking smartcards & RFID



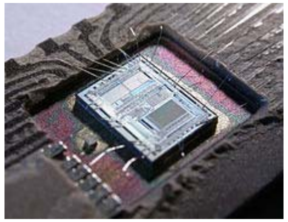
Erik Poll

Digital Security

Radboud University Nijmegen

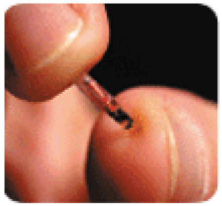
What are smartcards & RFID tags?

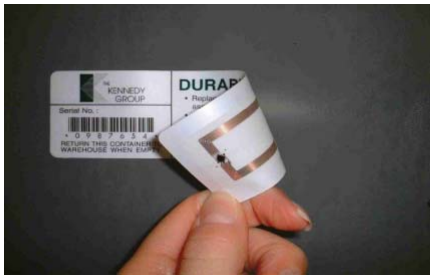
Micro-controller



with contact interface

or contactless interface





Why use them?

Convenience

- more convenient than username/password

Security

- more secure than username/password

Also more convenient & secure than barcodes and magstripes

What makes them secure?

- Tamper-resistant and tamper-evident – to some degree, but never tamper-proof

- no way to remove or access the "hard disk"

- therefore
  - any access to data - say the credit on your ov-chipcard - is under control of the card's functionality
  - the same goes for adding or changing code on the card
    - if possible at all

What can they do ?

1. stupid card just reports some data
   card shouts out a (unique) serial number on start-up

ISBN 0-596-00167-3
9 780596 001674

2. stupid smartcard aka memory card
   provides configurable file system with some access control
   - by means of *PIN code/passwords* or *crypto keys*
   - or even simpler: *irreversible writes (OTP or WORM memory)*

3. smart smartcard aka microprocessor card
   provides programmable CPU that can implement any functionality

Smartcard hardware for microprocessor cards

- CPU (usually 8 or 16, but now also 32 bit)
- possibly also
  - crypto co-processor & random number generator (RNG)
- memory: RAM and ROM & EEPROM
  - EEPROM serves as the smartcard's hard disk
- no power, no clock!

A modern card may have 512 bytes RAM, 16K ROM, 64K EEPROM and operate at 13.5 MHz

Do-it-Yourself

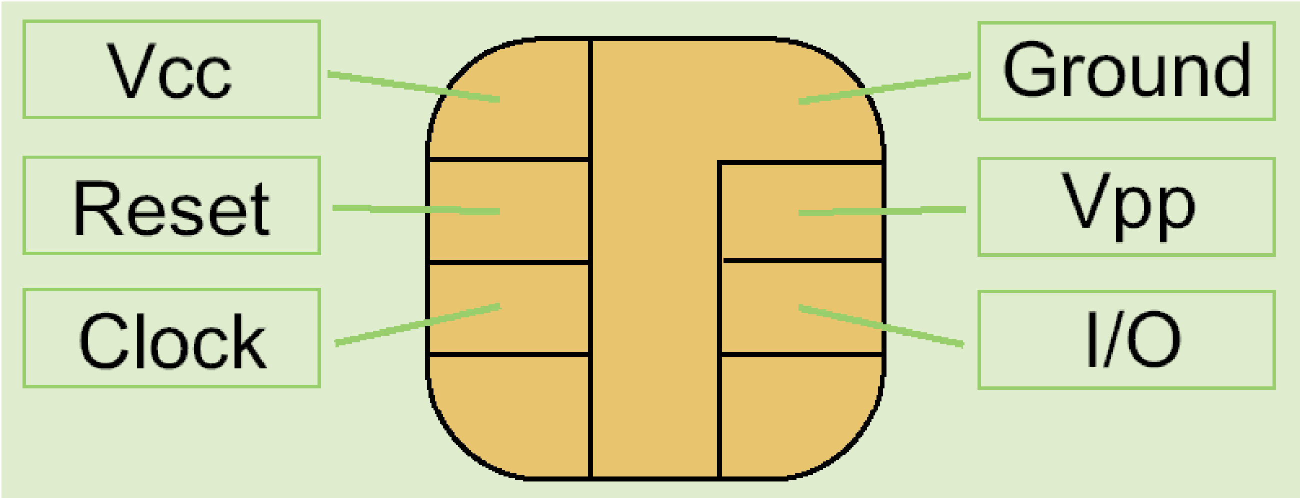
- Buy a card reader or NFC mobile phone
- Buy some tags and cards

- Programming you own smartcards is possible using JavaCard or MULTOS smartcards
- Check
  - www.ru.nl/ds/smartcards
  - libnfc
  - proxmark
  - rfidiot.org

Attacking smartcards and RFID

- logical attacks
  - find flaw in the functionality, targeting eg
    - the crypto – ie the cryptographic algorithms
    - the protocol
    - the key management
    - any other functionality
- physical attacks
  - physically mess with the card
- combinations
  - abuse functionality while you mess with the card

The simplest physical attack

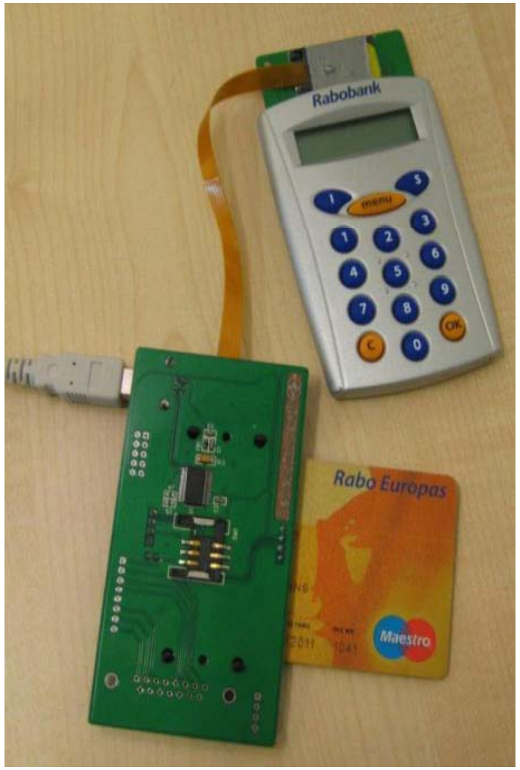


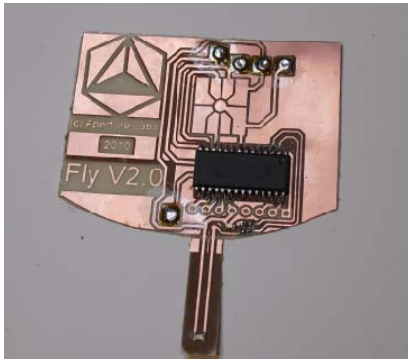
External power supply and external clock

- Vcc: orignally 5 V, now also 3V or 1.8V
- Vpp: higher voltage for writing EEPROM (13 V)

Vpp no longer used: painting over this contact is a major security threat

Logical attacks: tools of the trade



for passive eavesdropping or active Man-in-the-Middle

Logical attacks:
A very weak RFID tag

Mifare Ultralight

- Used in disposable ov-chipkaart
- No keys to protect memory access
- Relies on read-only and write-once memory for security
- Memory organised in 16 pages of 4 bytes
  - first part is read-only
    - includes 7 byte serial number
  - second part is One Time Programmable (OTP)
    - you can write 1's, not 0's
    - includes data for locking
  - third part is readable & writable

MIFARE Ultralight memory layout

Page	byte 0	byte 1	byte 2	byte 3
0	UID0	UID1	UID2	checksum
1				
2	checksum		lock 0	lock1
3	OTP 0	OTP 1	OTP 2	OTP 3
4				
5				
6				
7				
8				
9				
10				
11				

read only

read/ write

serial number UID

OTP

application data

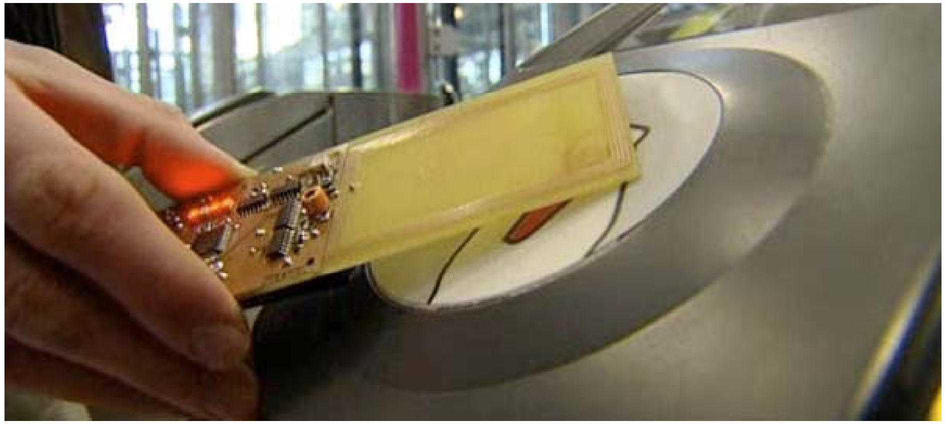
Flaw in disposable ov-chipcard

- wo lock bytes initially 0x00F0
- set to 0xF8FF to invalidate tag
- we can change an invalid tag so that terminals fail to recognize it as invalid...
- remaining 3 lock bits can still be set to one, so that lock bytes become 0xFFFF
- flaw in terminals: tags with lock bytes 0xF8FF are recognized as invalid, but tags with 0xFFFF are not
  - flaw since fixed by patching terminals

[Source "Security Evaluation of the disposable OV chipkaart", by UvA students Pieter Siekerman and Maurits van der Schee , July 2007]

More fundamental limitation: replay attack

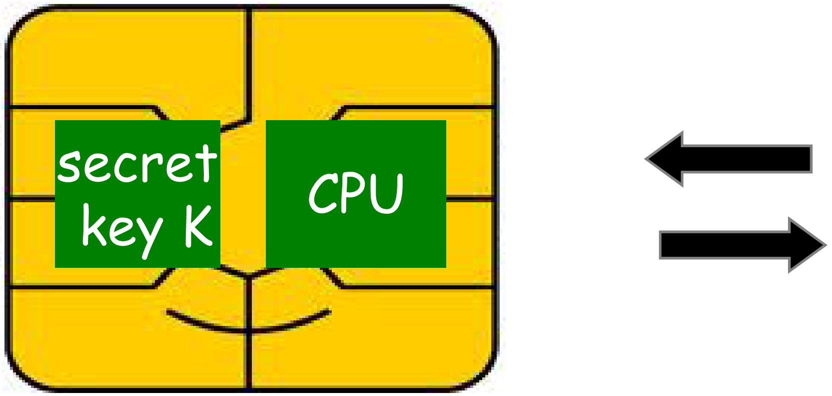
- Mifare Ultraright can store signed or encrypted data, but cannot do any processing, or offer any access control to reading the data
- *No* way to protect against spoofing of tags



- Only mitigation: serial number (UID) cannot be overwritten, so spoofing requires special hardware if UID is used

Logical attacks: Attacking the crypto

Challenge-response

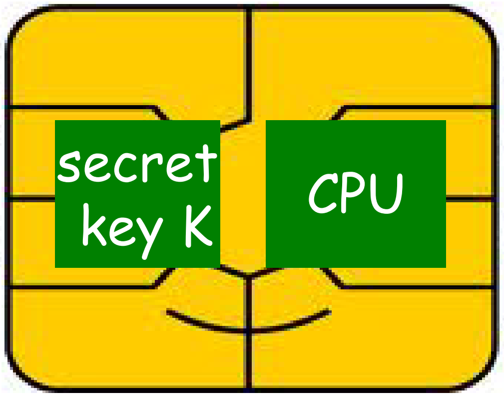


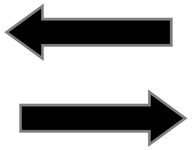
challenge c

response $\text{encrypt}_K(c)$

- If the card can do encryption, the secret key K never leaves the card
- Card issuer does not have to trust card holder, terminal, or network
- This is how you bank card works: it uses a 3DES key that only the bank knows

Breaking this?





1. Figuring out which encryption function is used
   - maybe this is known & published
   - otherwise: reverse engineering, experimenting to figure out how encryption works
2. For poor encryption: by trying out few challenges, you may be able to reconstruct key

For good crypto – 3DES, AES, RSA,... – this is hopeless

Proprietary crypto broken in DS group

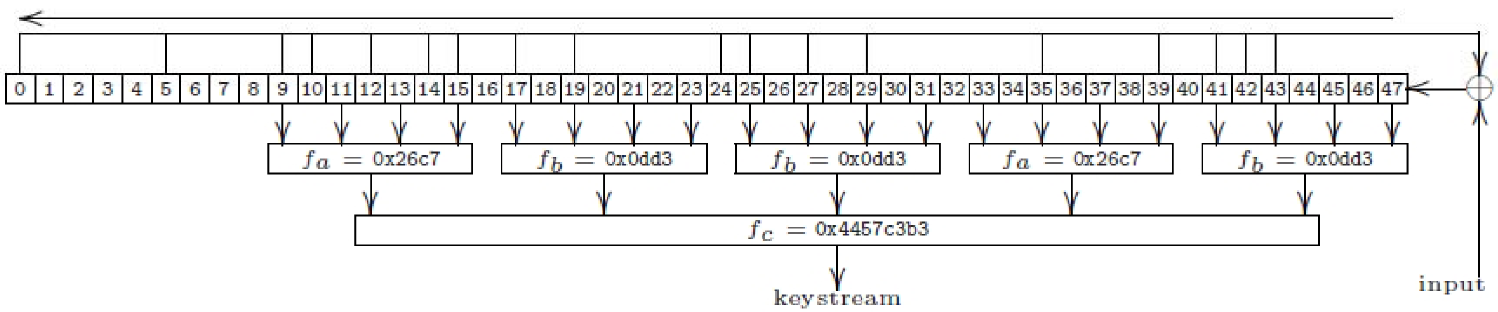
- Mifare Classic

- ATMEL SecureMemory, CryptoMemory and CryptoRF
- HID iClass and iClass Elite

- Hitag2

- Moral of the story: use established, crypto primitives
  - publicly studied according to Kerckhoffs principle

Crypto 1 in Mifare Classic



Logical attacks:
Attacking the key management

Common problems with crypto keys

- people using the same key in all cards
  - for one customer, or - worse - all their customers!
  - HID iClass uses a globally unique master key, which is built into all HID card readers
- worse still, using the default keys
  - **75%** of MIFARE applications was found to use default keys or keys used in examples in documentation

    [Source: Lukas Grunwald, DEFCON14, 2007]
  - A0A1A2A3A4A5 is an initial transport key of MIFARE tags. Googling for A0A1A2A3A4A5 produces links to documentation with other example keys to try!

Logical attacks:
attacking security protocols

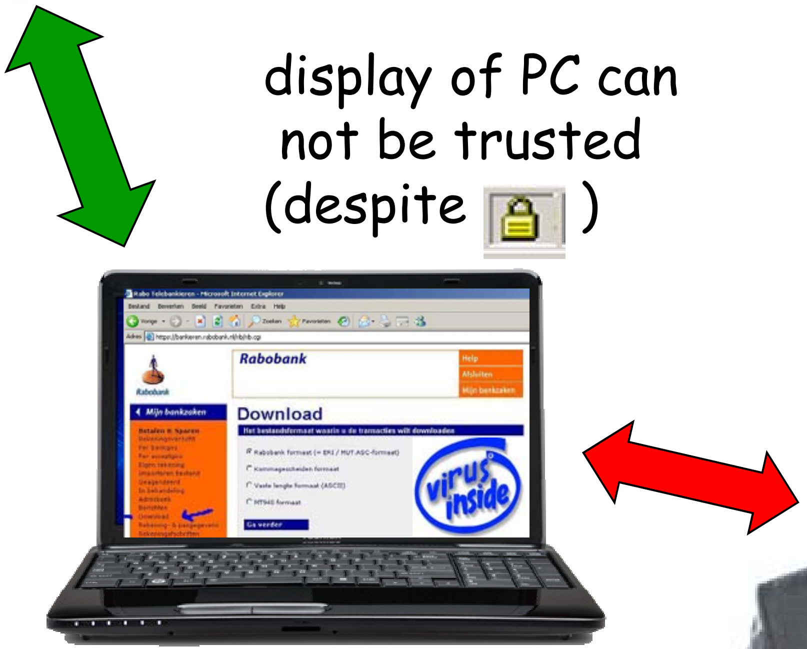
Fraud with internet banking in Netherlands

2008	2.1 M€
2009	1.9 M€
2010	9.8 M€ (7100€ per incident)
2011	35 M€ (4500€ per incident)
2012 (1st half)	27.3 M€

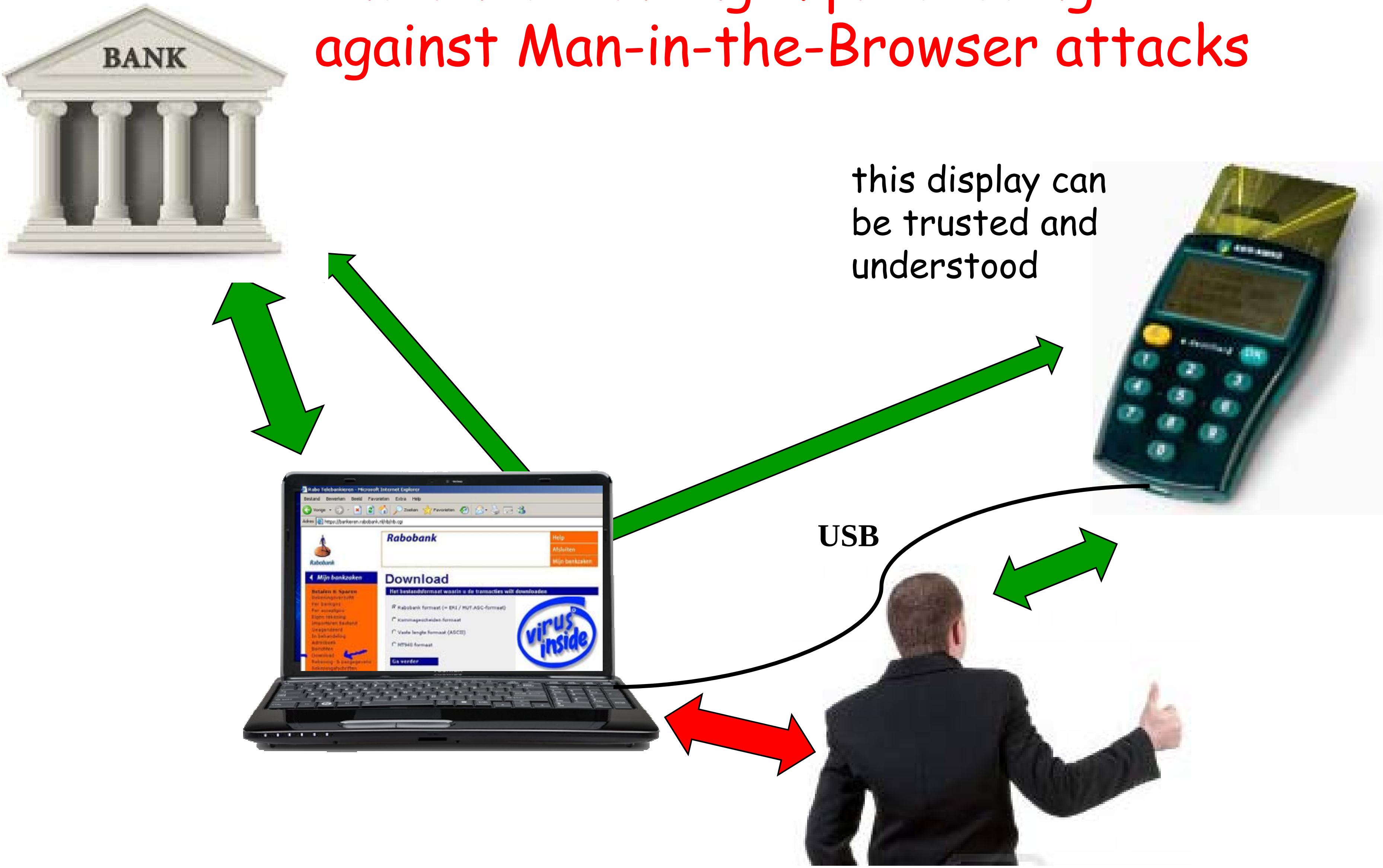
[source: NVB]

Internet banking & Man-in-the-Browser attacks

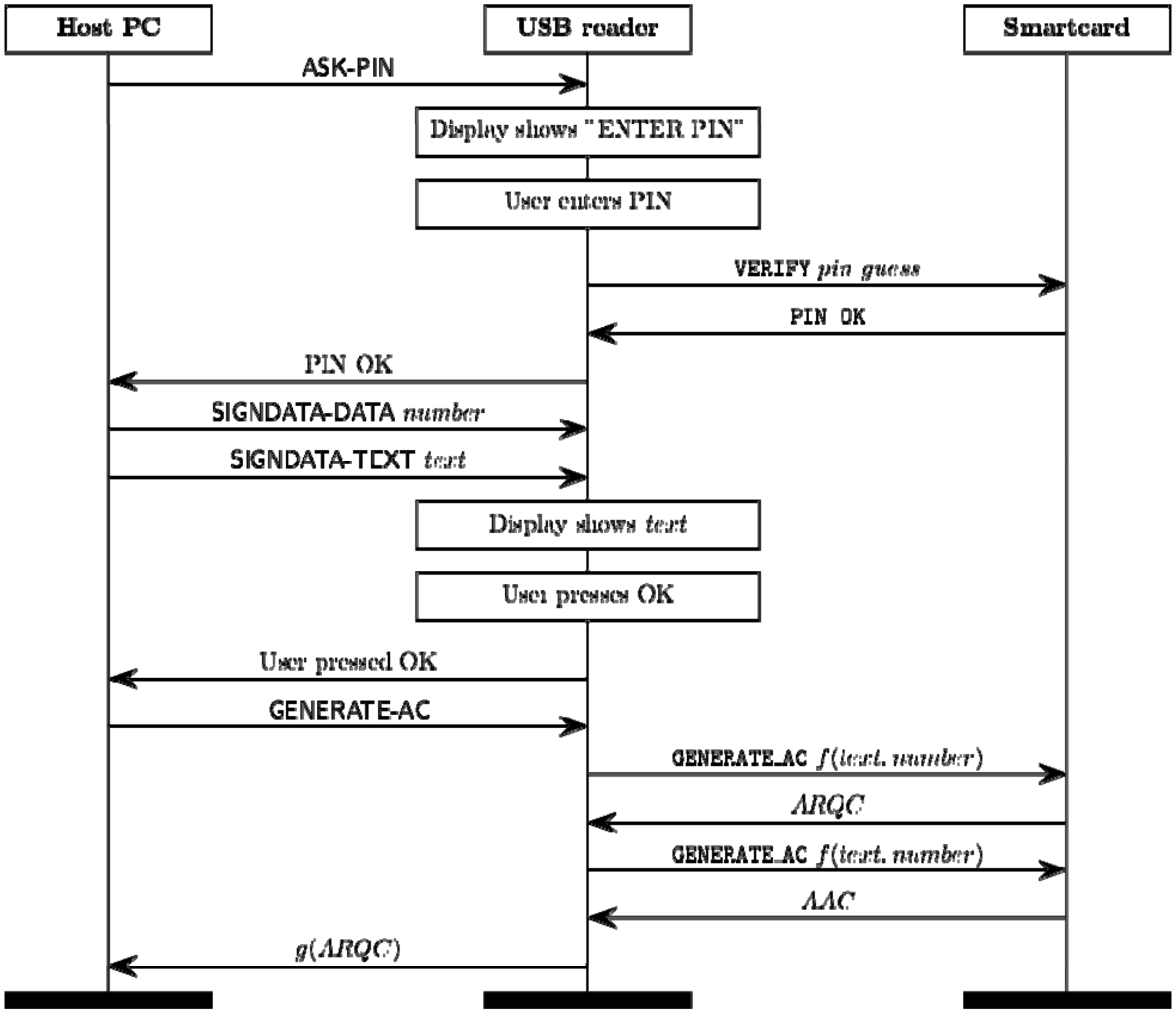
display of PC can not be trusted (despite)



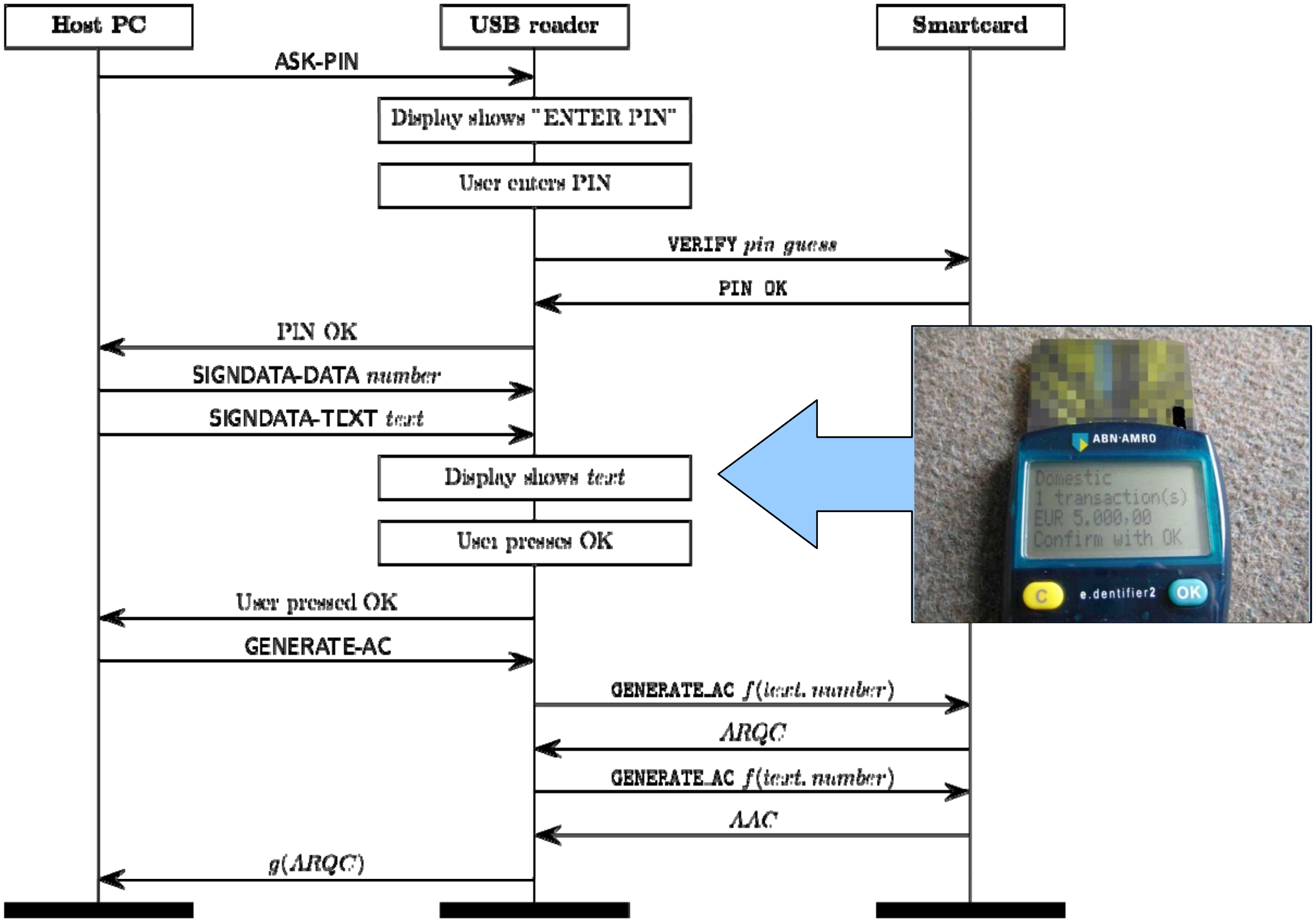
Internet banking & protecting against Man-in-the-Browser attacks



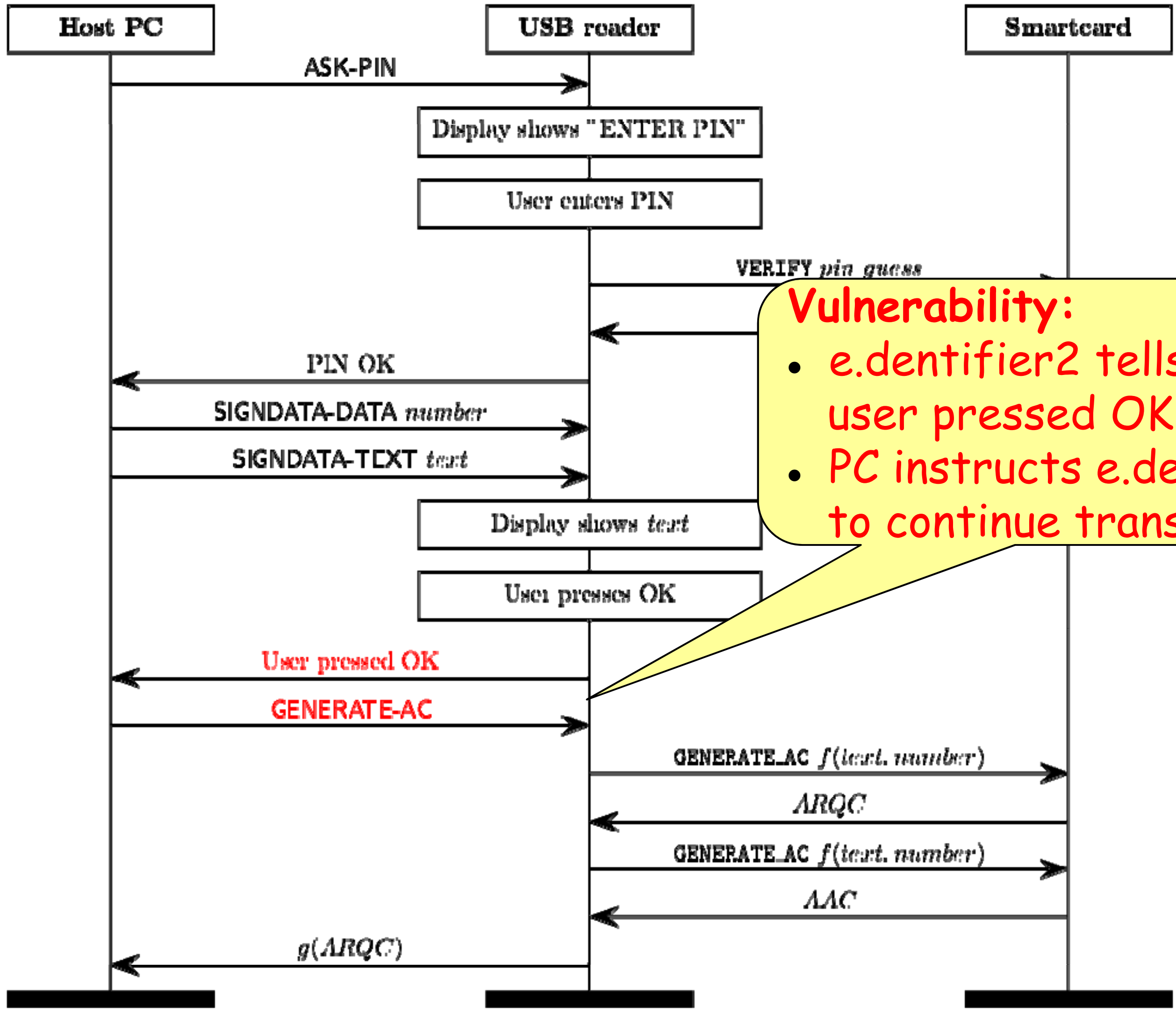
Protocol of USB-connected e.dentifier2



Protocol of USB-connected e.dentifier2



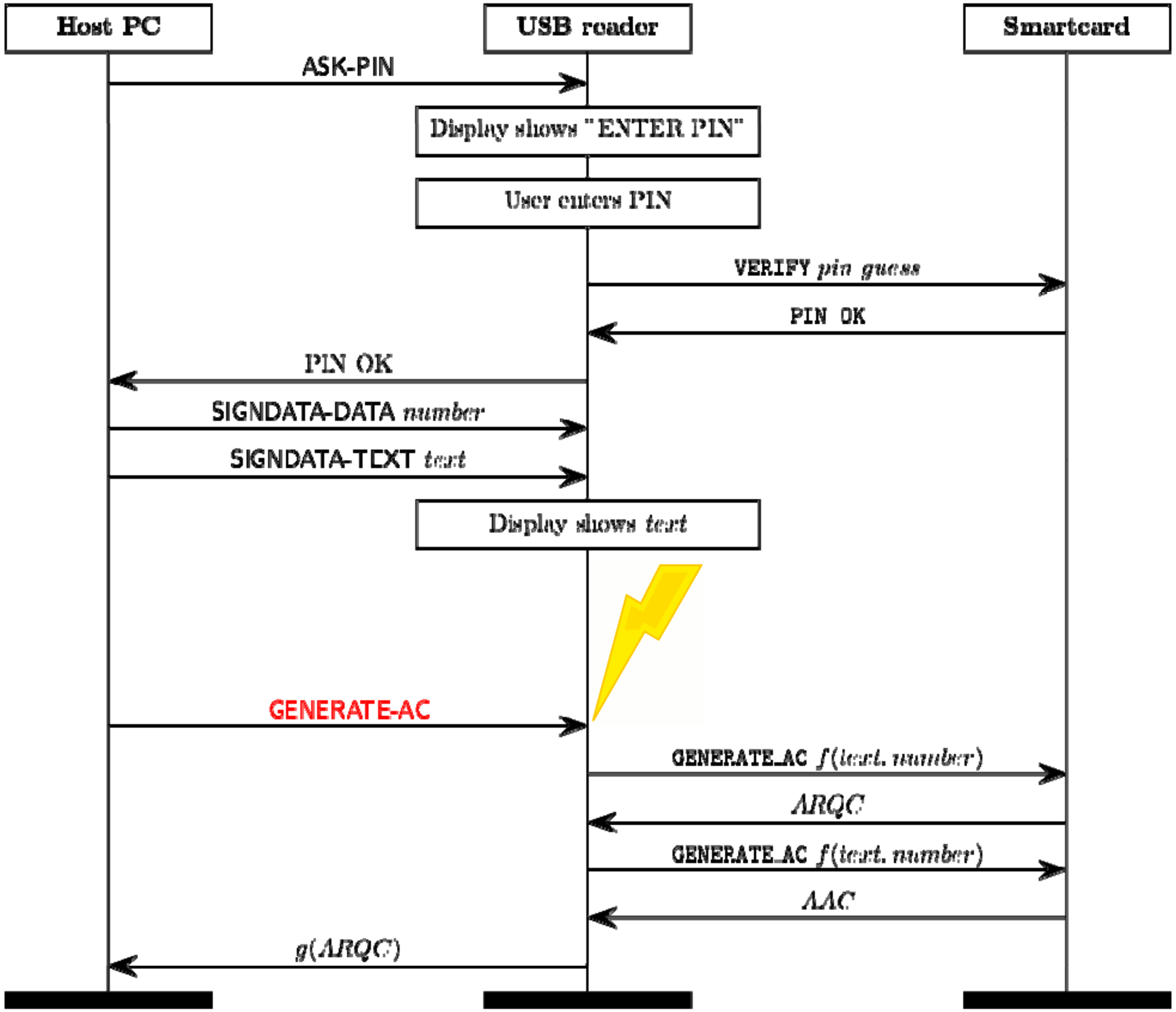
Protocol of USB-connected e.dentifier2



Vulnerability:

- e.dentifier2 tells PC that user pressed OK
- PC instructs e.dentifier2 to continue transaction

Attack



Host PC
USB reader
Smartcard
ASK-PIN
Display shows "ENTER PIN"
User enters PIN
VERIFY pin guess
PIN OK
PIN OK
SIGNDATA-DATA number
SIGNDATA-TEXT text
Display shows text
GENERATE-AC
GENERATE_AC f(text, number)
ARQC
GENERATE_AC f(text, number)
AAC
g(ARQC)

Movie


U bent ingelogd.
Laatste bezoek:
26 okt 11, 14:09u
ABN·AMRO
e.dentifier2
Maestro
Wereldpas

Other example logical weaknesses
-
for e-passports

Unwanted functionality

- Test version of Dutch passport
  provided software emulation of Mifare Classic

- with default key, of course...

This allows adding a cloned ov-chipcard on the passport

Attacking the terminal software

- Lukas Grunwald managed to crash e-passport terminals by sending a malformed JPEG
  - causing a buffer overflow in the graphics library

- *Smartcards and RFID tags should be treated as untrusted inputs*
  - *until we have authenticated the card and/or the data it provides*

e-passport leaking info by error response

	2 byte error response	meaning
Belgian	6986	not allowed
Dutch	6982	security status not satisfied
French	6F00	no precise diagnosis
Italian	6D00	not supported
German	6700	wrong length

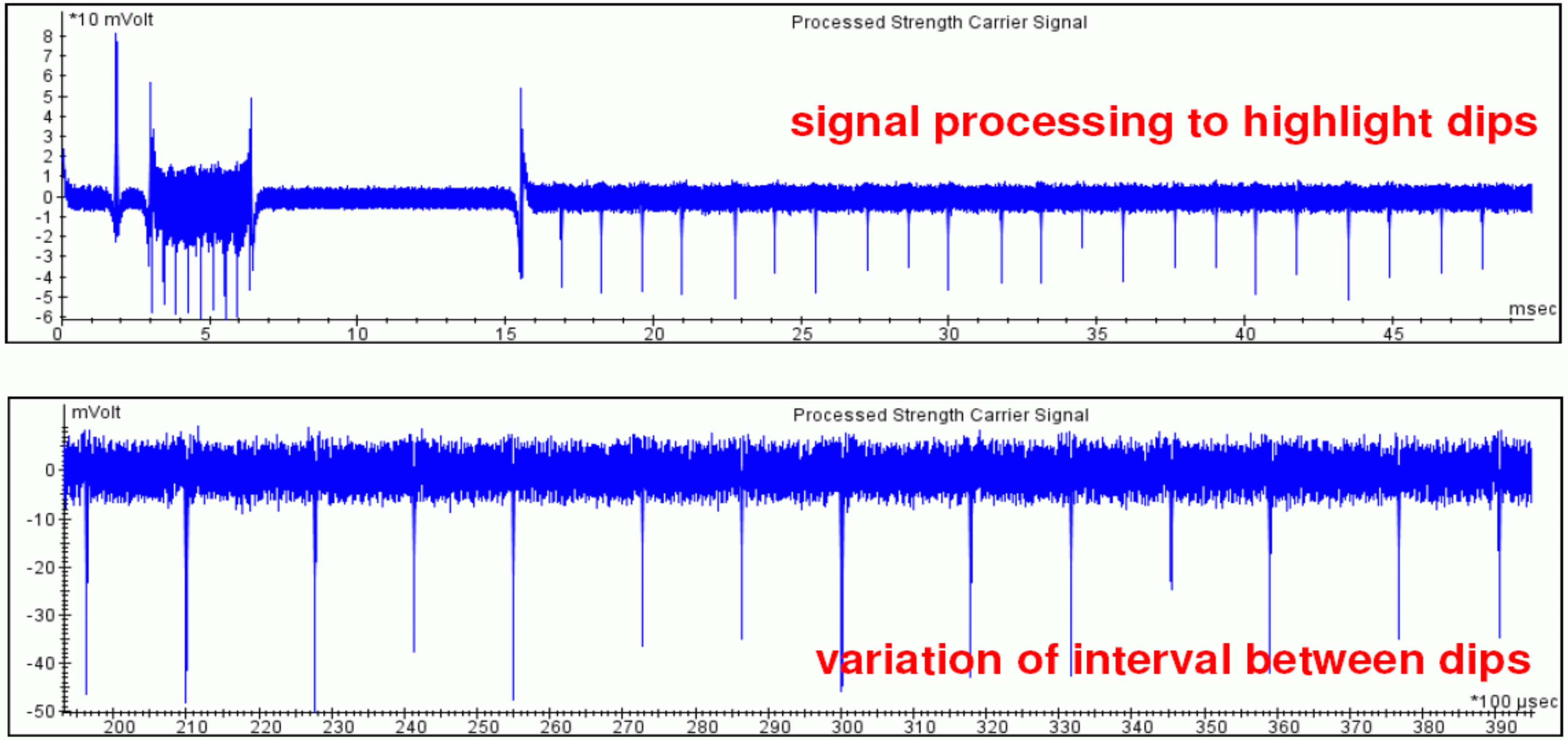
Error code for illegal B0, ie. READ BINARY, instruction

This reveals the nationality of a passport
- in spite of access control to passport data

But attack range limited to 30 cm, so danger of passport bombs overhyped

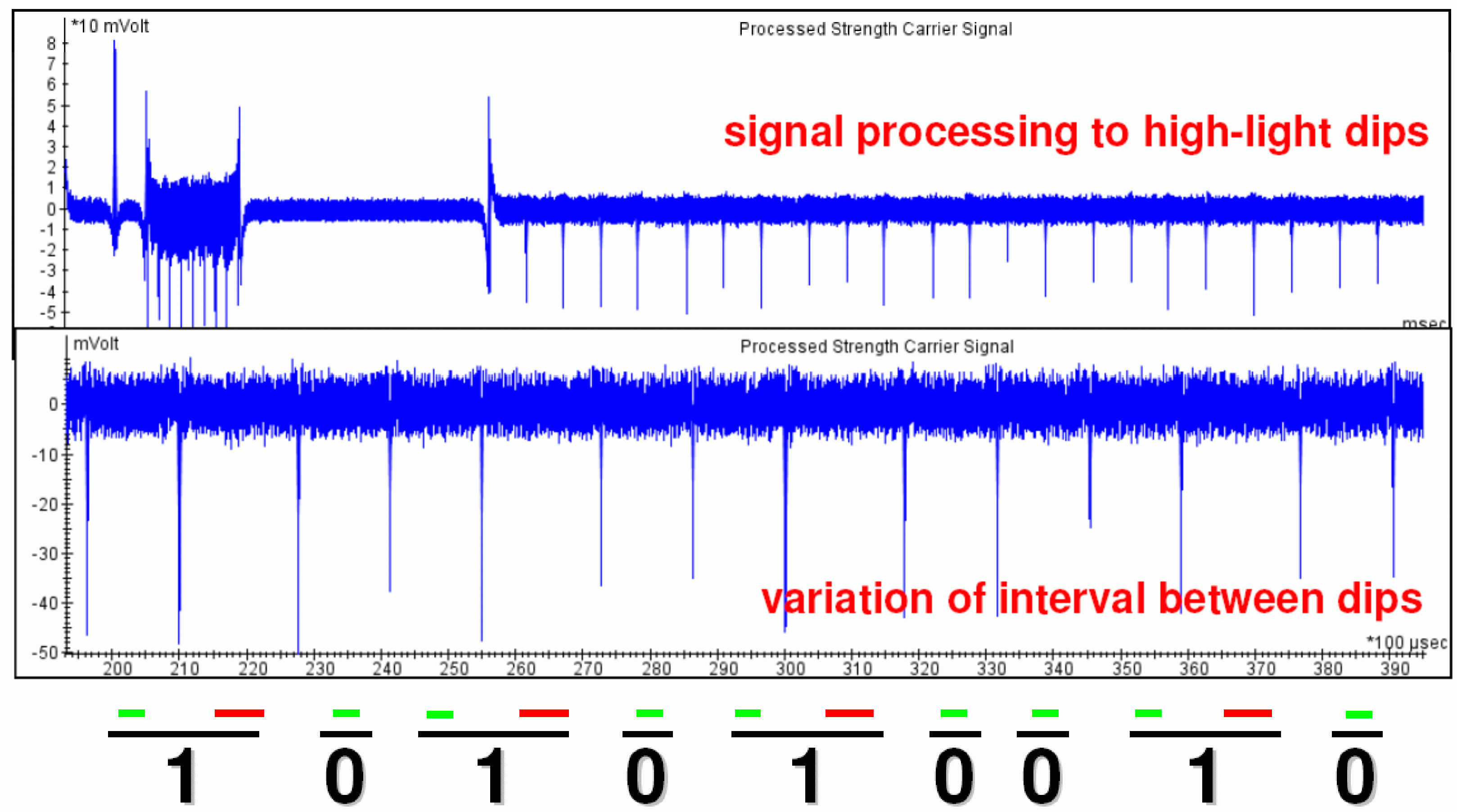
Physical attacks:
side-channel attacks

Power trace of an RSA encryption



[Source: Riscure]

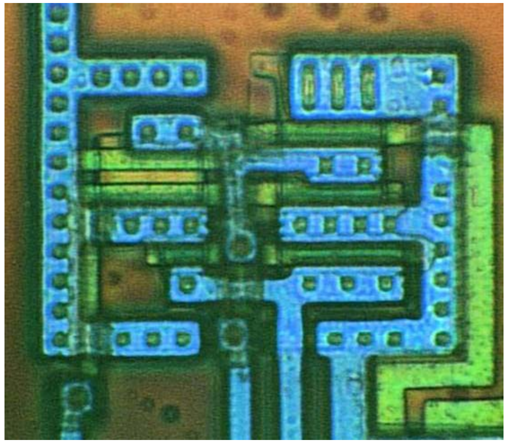
Power analysis: reading the key from this trace!

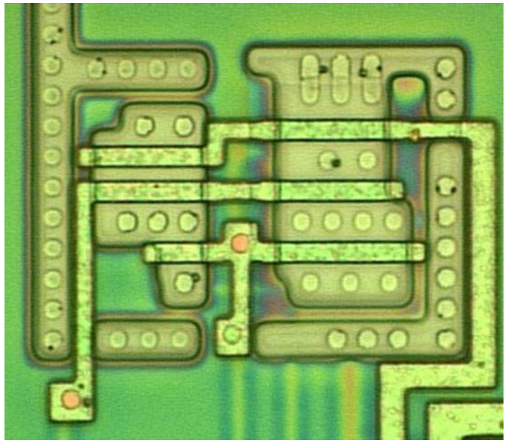


Physical, invasive attacks

First step: removing chip from smartcard

Optical reverse engineering

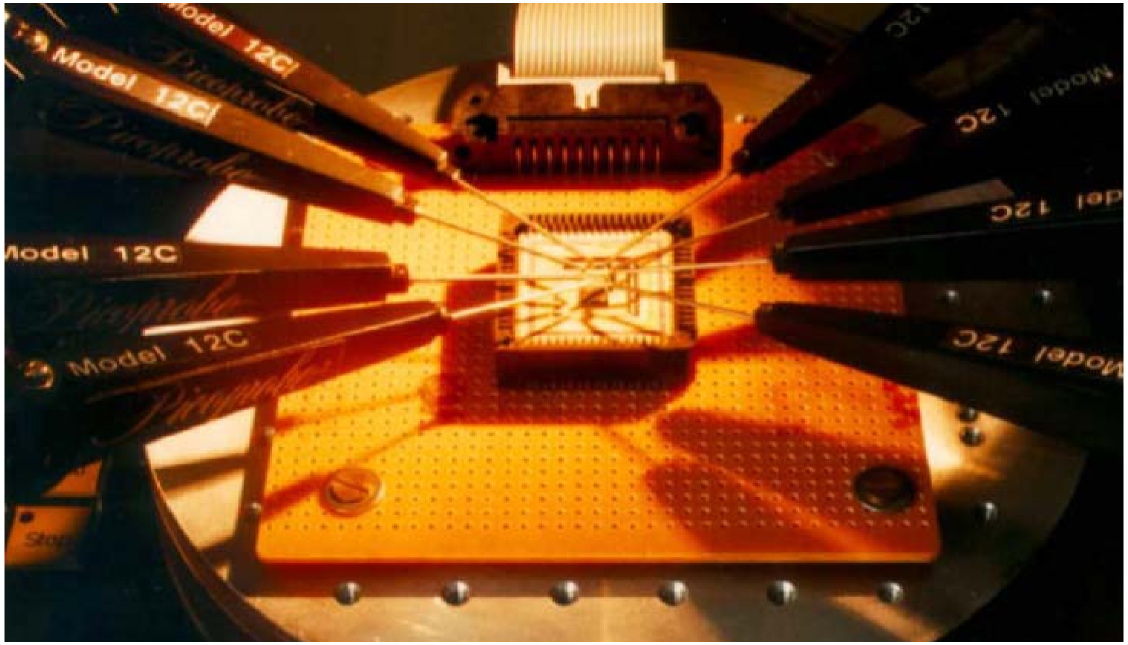




Probing

- Observe or change the data on the bus while the chip is in operation.

  eg to observe key

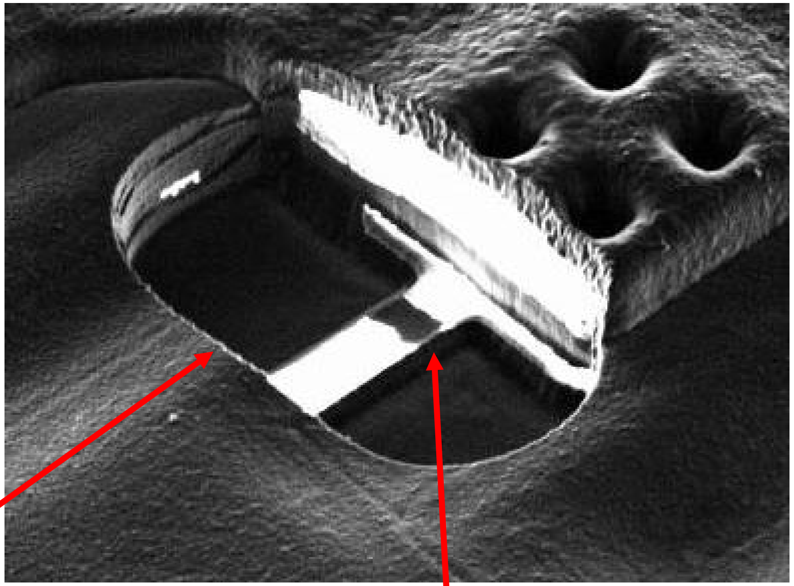


Fibbing

FIB = Focussed Ion Beam

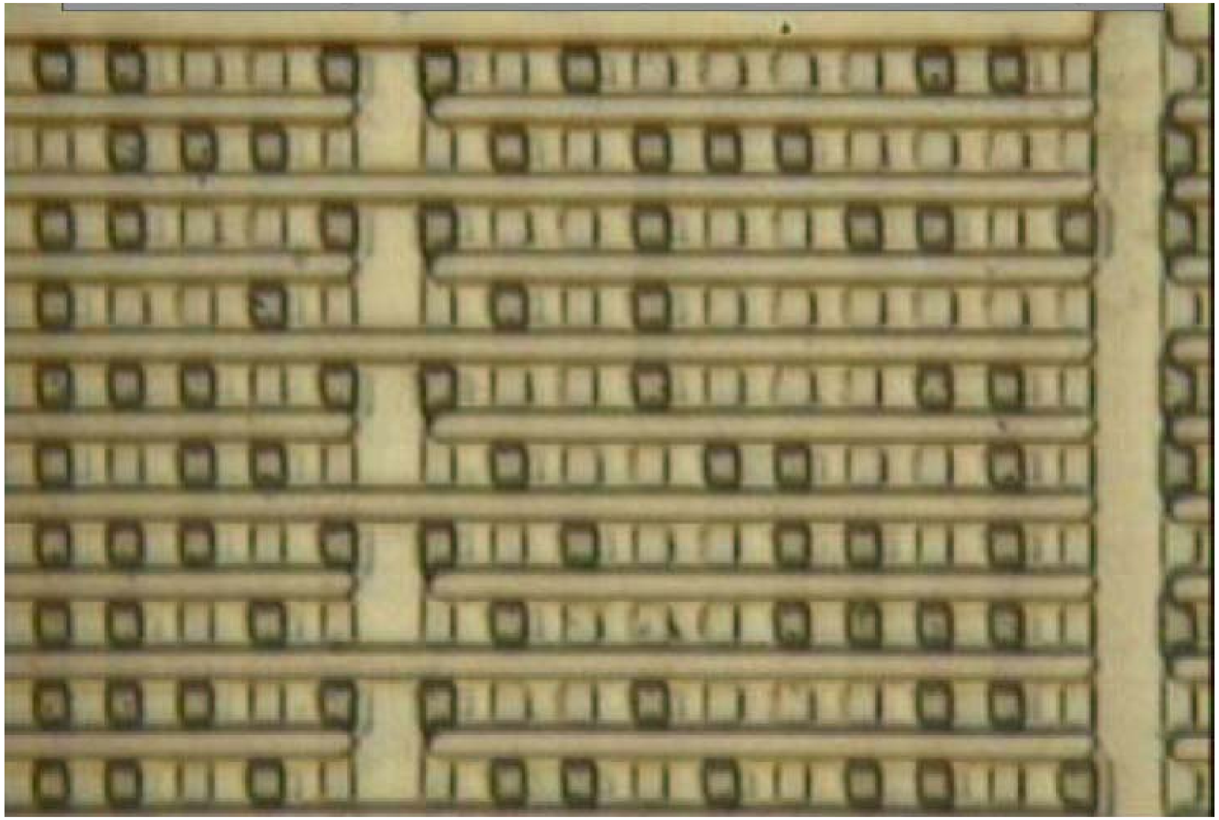
can observe or modify chip by

- drilling holes
- cutting connections
- soldering new connections and creating new gates



Extracting ROM content

Staining can optically reveal the bits stored in ROM:
dark squares are 1
light squares are 0



[Source: Brightsight]

Latest fashion: fault attacks

- Introduce a fault while chip is operating
  - by glitching: dipping the voltage
  - by shooting a laser at the chip

Conclusions

- Smartcard & RFID security not perfect
  - cheap, logical attacks
    - little equipment, but some time & brainpower
  - expensive, physical atacks
    - more equipment
  - both can be devastating...
- The ongoing arms race between defenders and attackers will never end
  - these days esp. for side-channel and fault attacks