1st panick

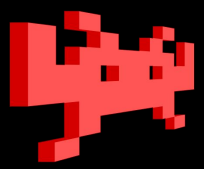
GreHack

2012 WEEK 42

Broadcasting encryption or systematic #FAIL ?

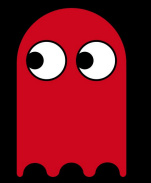



Phil




BEWARE
THE
INVASION



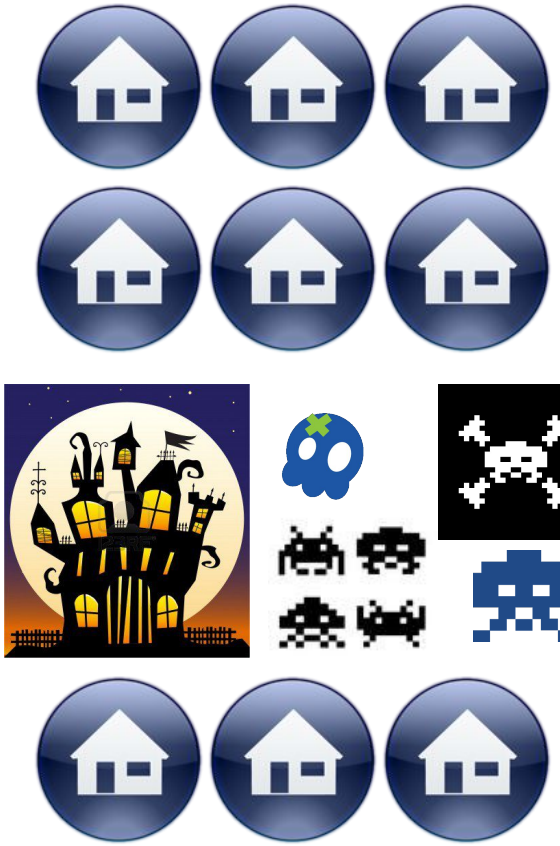
SUMMARY

- Intro : Broadcasting something...
- 1984 : Discret 11
- 1995 : Syster
- 1996 - 2002 : Seca 1
- 2002 - 2008 : Seca 2
- Conclusion

Broadcasting : For the masses

Encrypted Stream

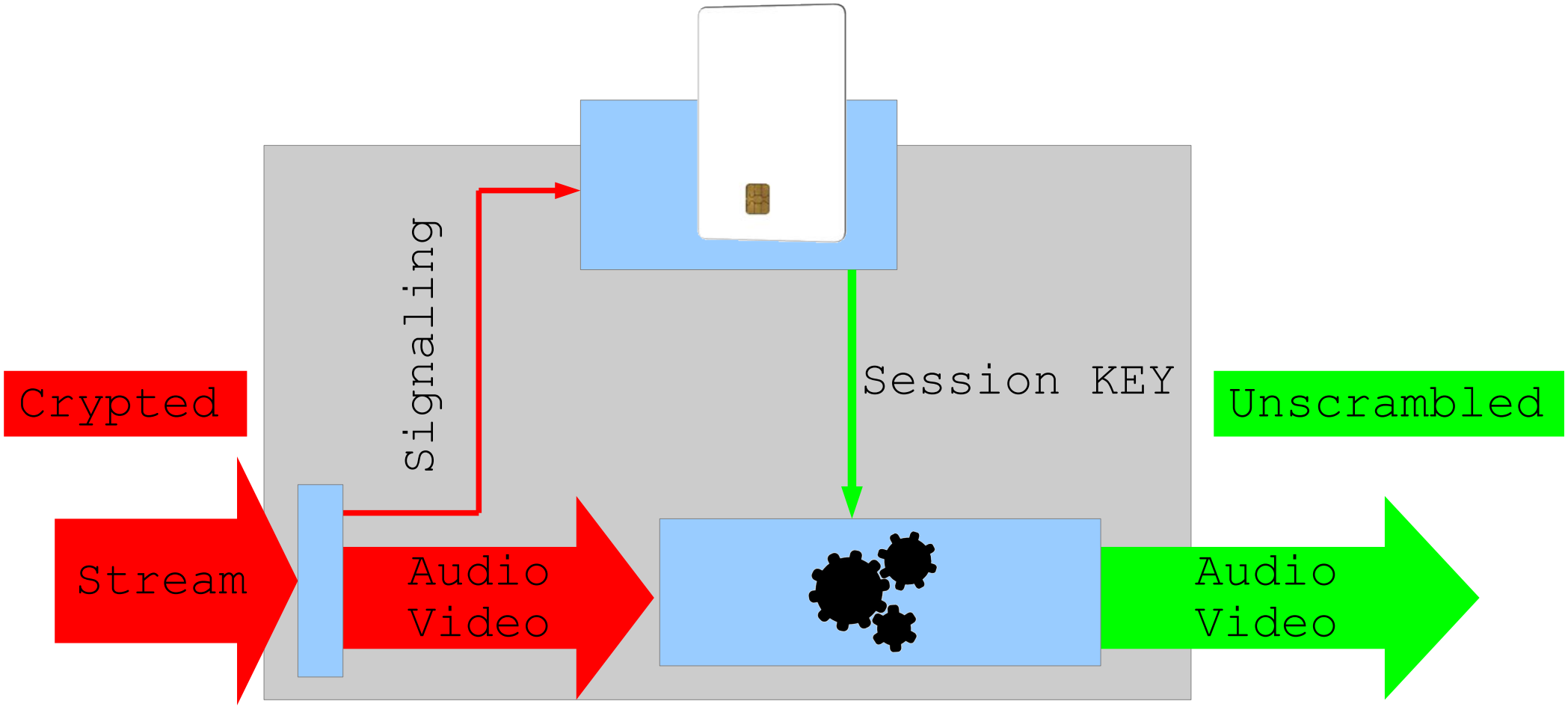
- Picture + sound
- Signaling



2 problems :

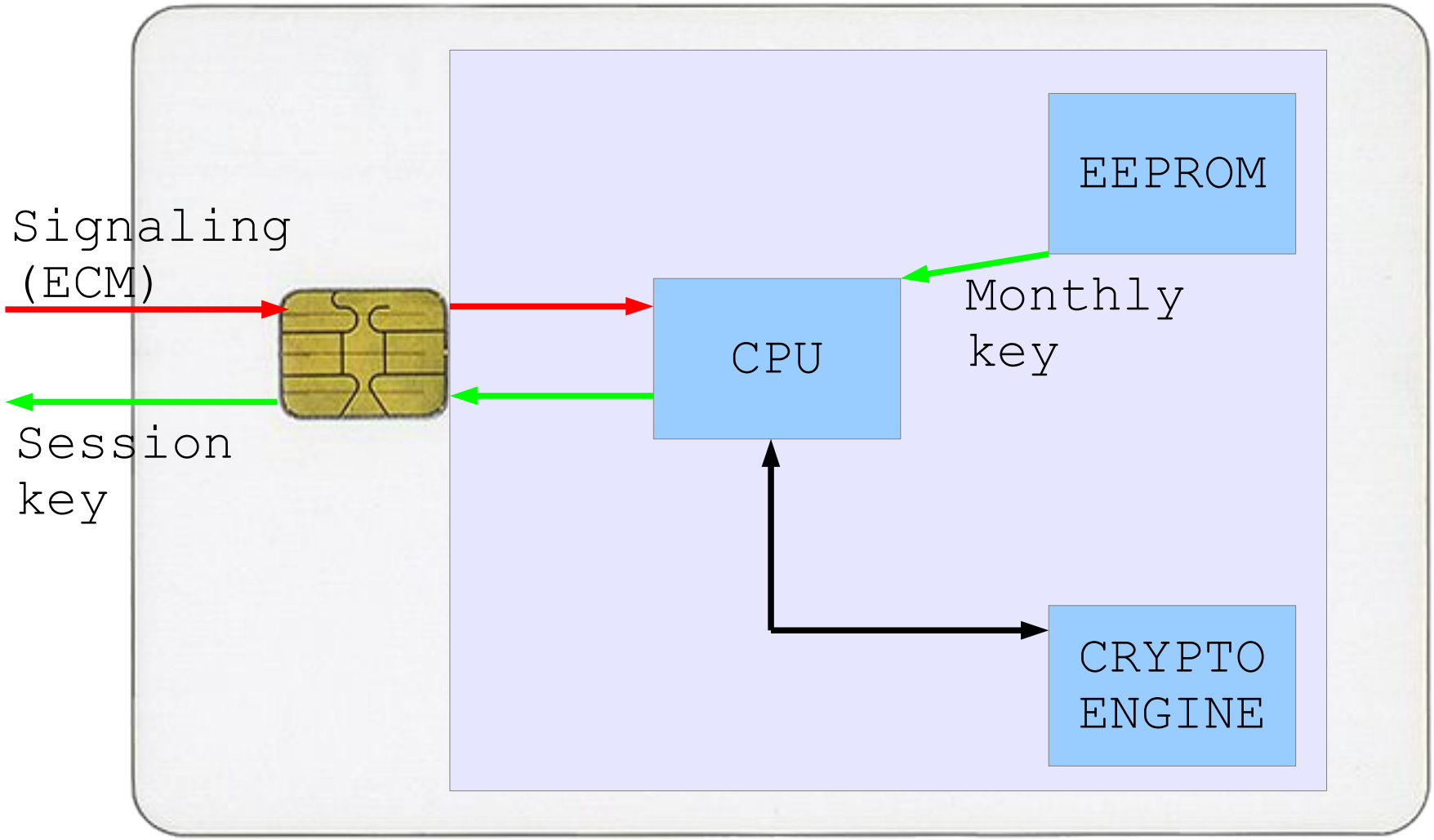
- Broadcast is for everyone
- No uplink

Broadcasting : The receiver



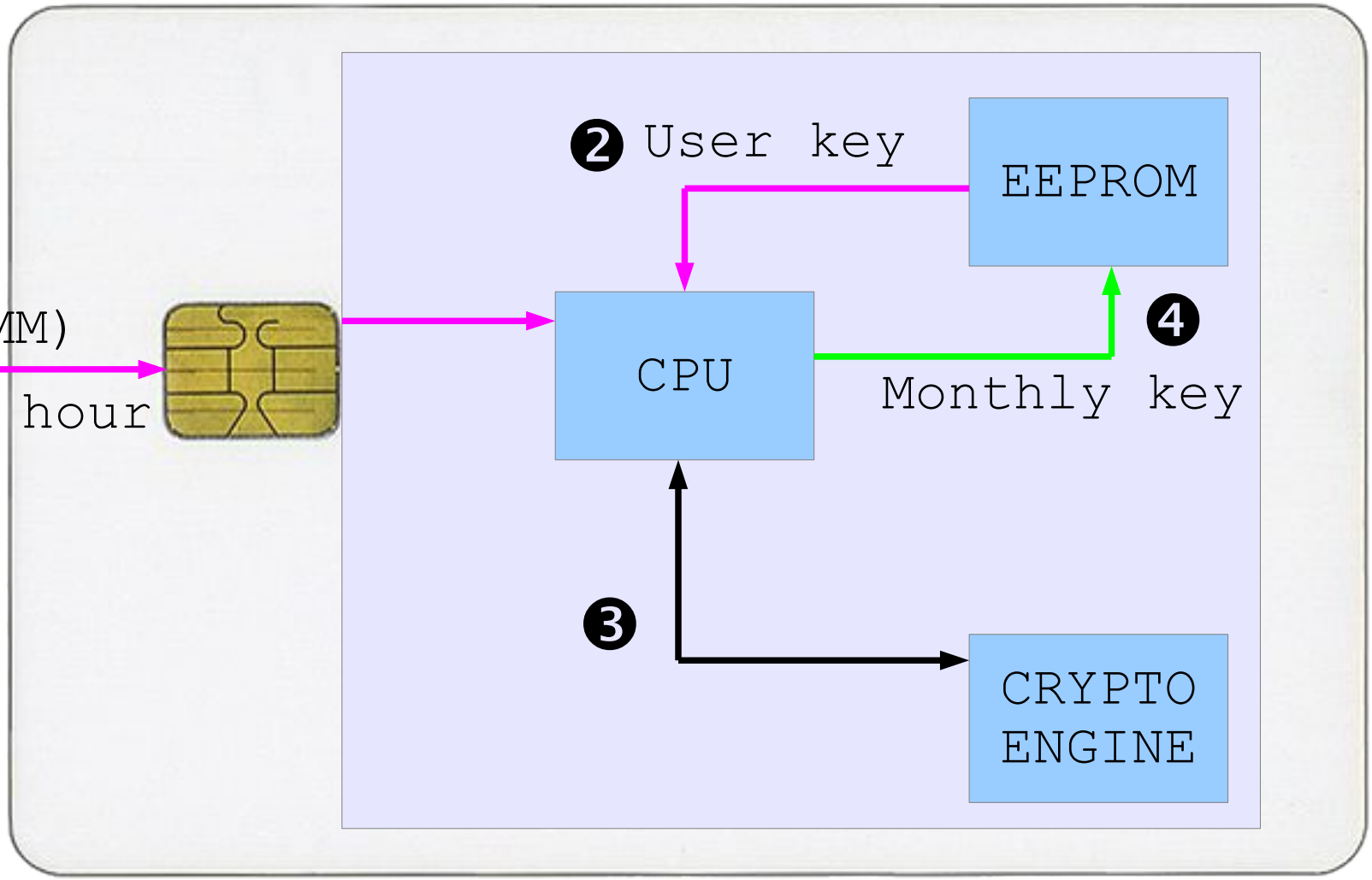
Bad luck : RED and GREEN are UNIQUE for EVERYONE

Broadcasting: Secure device



Bad luck[2] : RED and GREEN are UNIQUE for EVERYONE

Broadcasting : Monthly update



The UNIQUE value per subscriber is an UPDATE key !

1984

The first pay-TV system in France :

« DISCRET 11 »

Discret 11

Discret 11

Vintage !

Discret 11

Main motivation ?

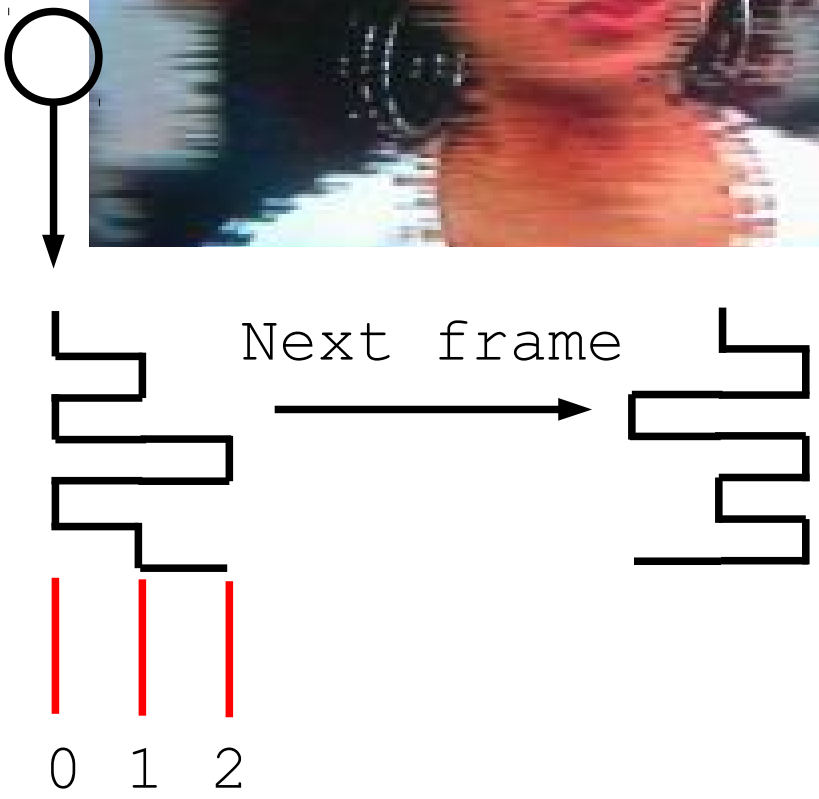
- Challenge
- Fun
- $$$
- And …

Discret 11

A porn movie broadcasted each first Saturday of the month …

Discret 11 : The image

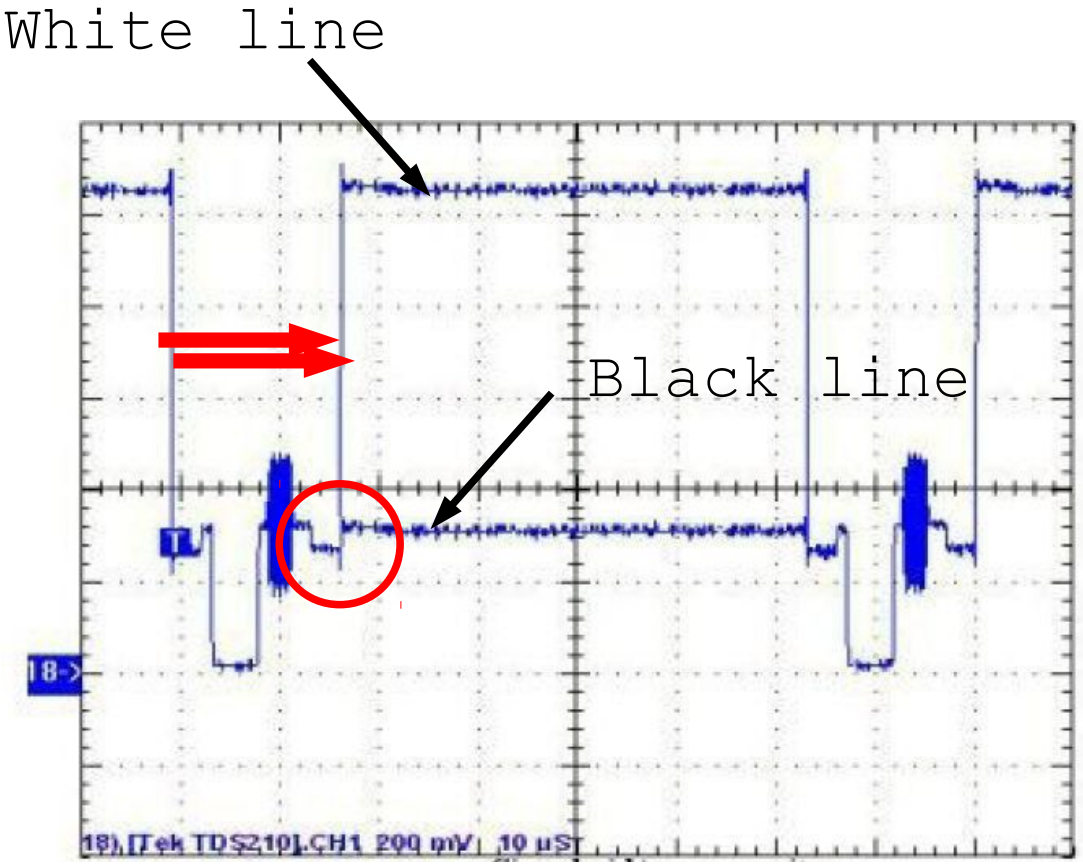
The image, regarding to the timing aspect of each line.



3 differents lines
0 : need to wait 1804ns
1 : need to wait 902ns
2 : line on time

Discret 11 : First hack

First implementation

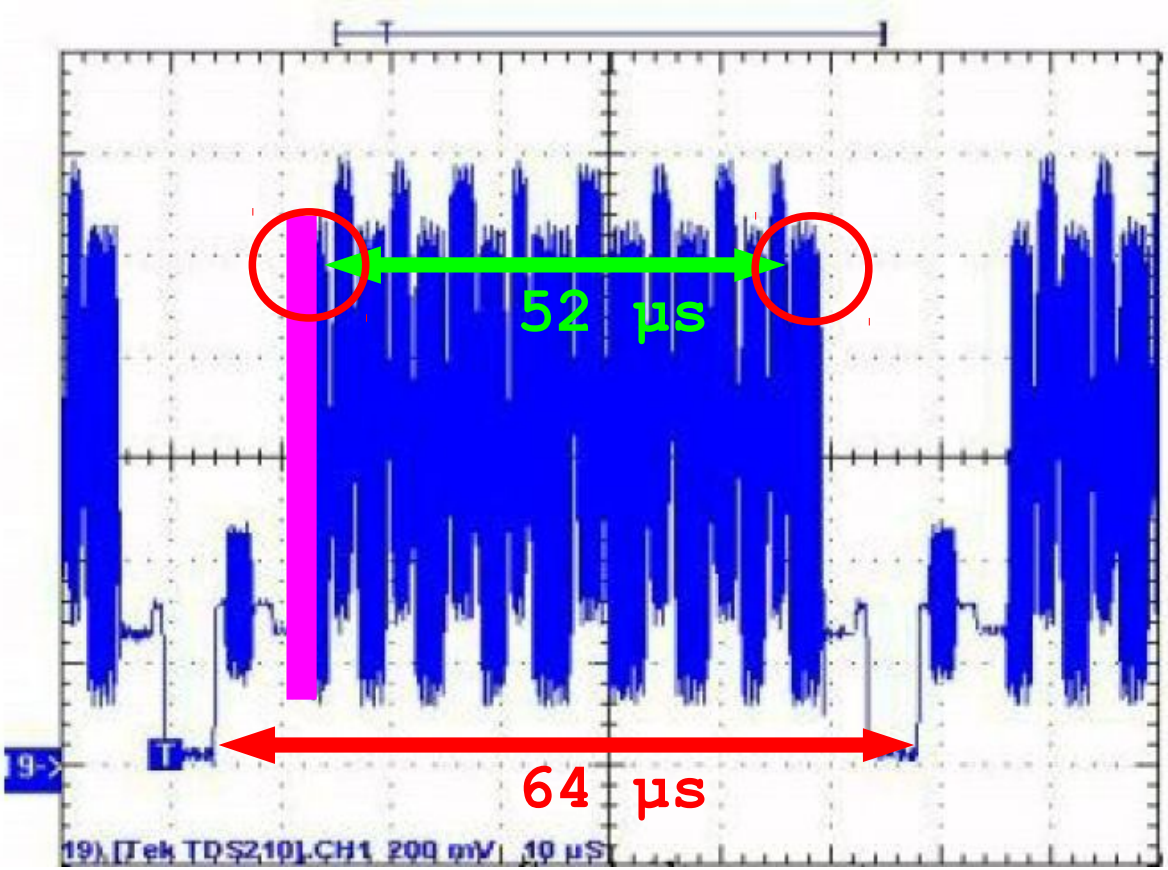


Voltage comparators, and live line timing adaptation.

Broadcast start the 4 November 1984.
This implementation was published in December 1984 in a french electronic magazine !

Not a perfect solution, sometime with black line, the hack didn't apply the correct delay.

Discret 11 : countermeasures



Countermeasures to kill pirate decoder :

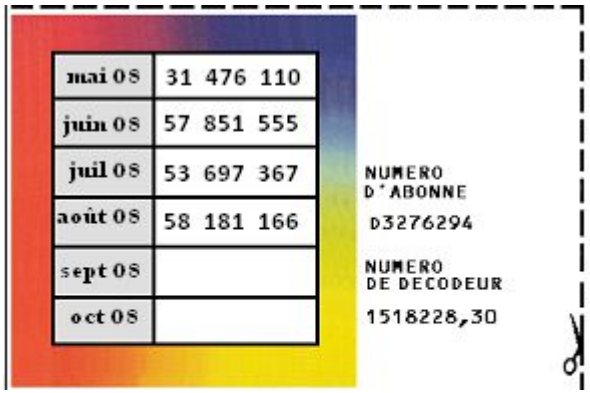
All the line that must be late are cropped.

As only 52µs of signal is displayed, it did not disturb too much the viewer.

Discret 11 : clone

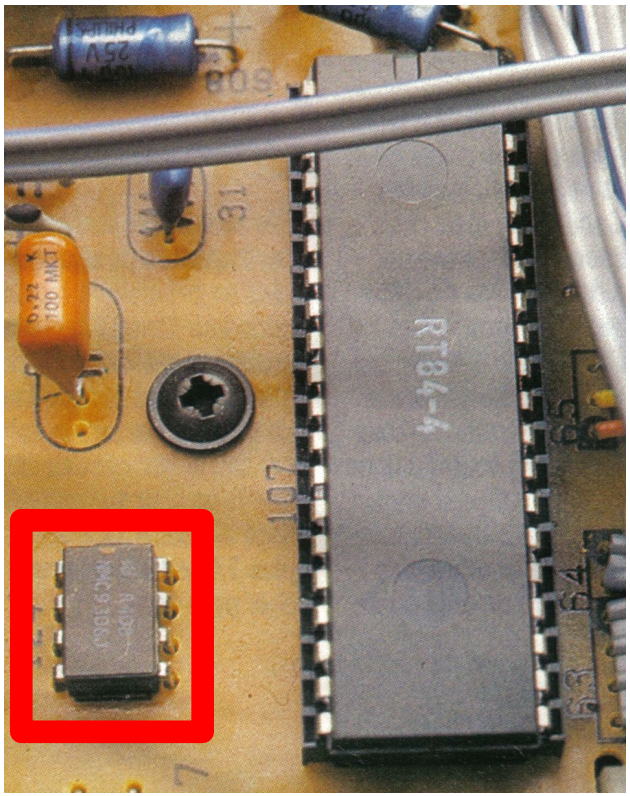
The EEPROM to clone for using the subscription of your friend. In 1984 it was secure, because no one own an EEPROM programmer ☺

Monthly update coupon



Dump of the EEPROM 9306

```
FFFF FFFF
A1AA A1AA
A2AA A2AA
FFFF A305
A405 FFFF
FFFF A566
A6A8 A6AC
FFFF FFFF
```



Discret 11 : full disclosure

The full encryption scheme was discover by only looking the sequences of pictures.

- Each 6 frames the scrambling sequence restart
- 11 for 2^11 : the size of the delay table
- The monthly code is only the start point of the delay table.
- Ability to automatically find the monthly code.

Important : today the original code in the official receiver have never been dumped !

Nothing other than pictures was needed to break the system.

1995

Second pay-TV
system in France :

« SYSTER »

Syster : the decoder

All the secret in the white KEY but ...

Syster : the picture

Syster : picture more closer

With SECAM one line have RED color information encoded, the next one the BLUE, and so.

The line MIX done by the crypt system break this rules of RBRBRBRBRB… lines.

While monitoring the color of each lines it was possible to reconstruct each frame, without knonwing the real algorithm & session KEY.

The decoder was build by Kudelski, but we add always bet it was designed for PAL system.
Bad luck, it's SECAM in France …

They did not learn from the 1984 lesson!

Syster : bad dayz, but ...

The countermeasure had certainly ask a lot of brain at Kudelski engineer : find an encryption scheme who did not break the RBRBRBRB… sequence.

They manage in and kill all the pirate decoder. One point !

But …

The white KEY as show all it's secret with a simple buffer overflow & code injection.

Then, dump, learn, implement & so …

1996-2002

Satellite broadcasting,
Stage 1 ...

« SECA 1 »

SECA 1

First try for the provider in the wonderful satellite world with European broadcasting.

All the cards broken with software bugs.

All secrets known, thousands implementation of emulators.
Revival and clone of official cards without much pain.

All the different version of SECA 1's ROM had never secure anything more than a short period …

2002-2008

Satellite broadcasting,
Stage 2,

« SECA 2 »

SECA 2 : The black screen

This time, a secure card ...

Let's share it !

SECA 2 : Share your card

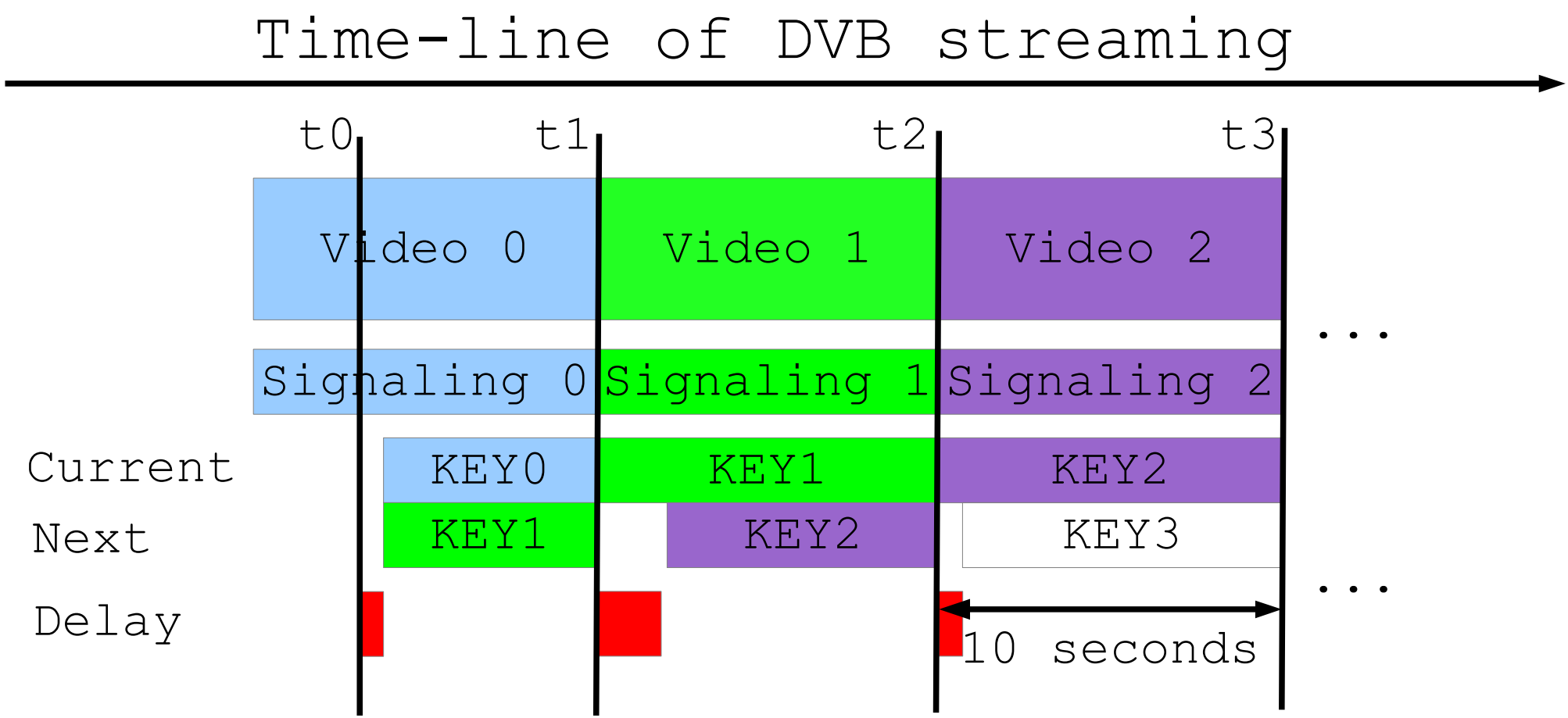
DVB norm born in 1993 :

No connection between peers in 1993,

but now-day it's a fact.

Let's exploit the bad design !

SECA 2 : the DVB flows

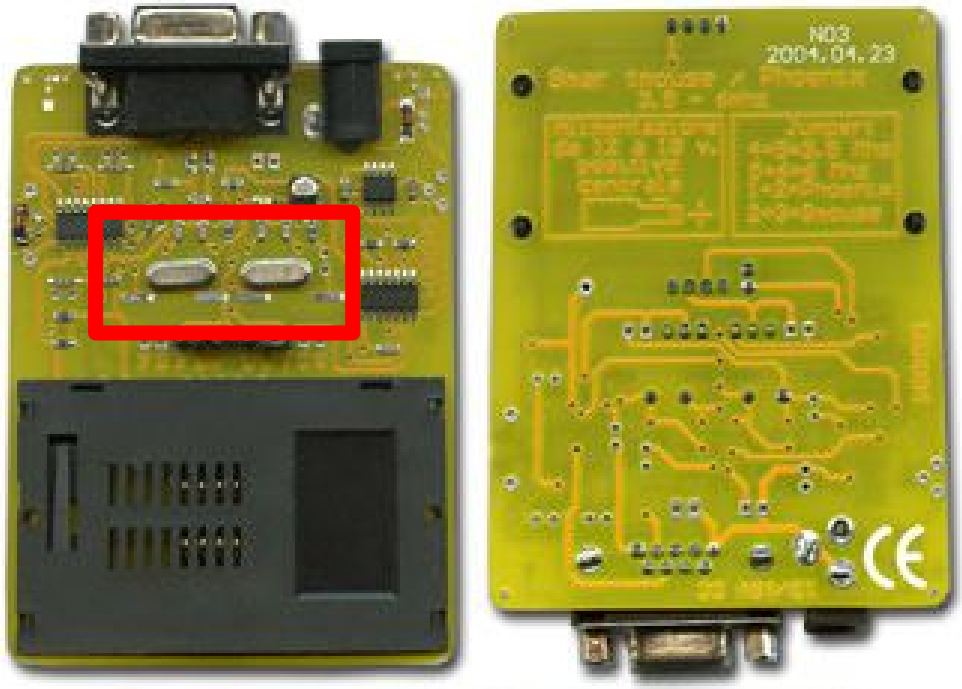


t0 : zap on a channel, ask KEY0|KEY1, use KEY0
t1 : use KEY1 and ask for KEY1|KEY2

10 secs to grab the next session key, an eternity ...

SECA 2 : hardware for sharing

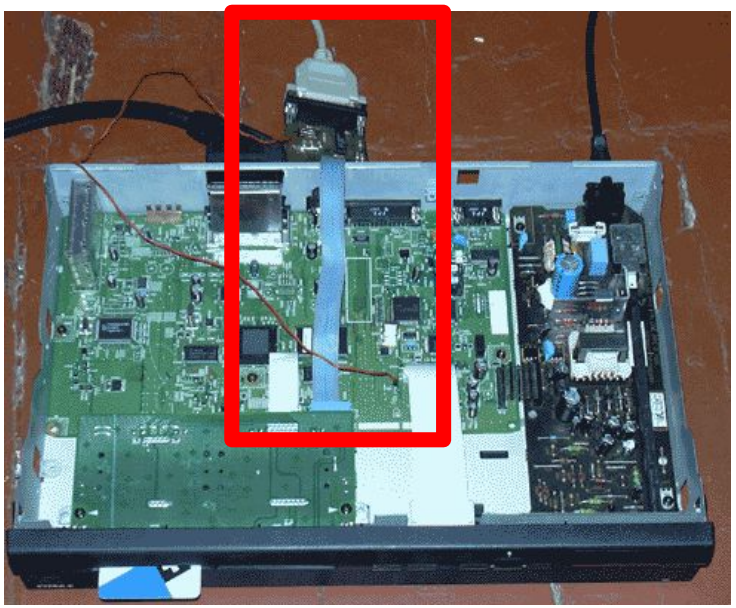
Server side : a cheap ISO7816 smarcard reader and a TCP/IP session manager.

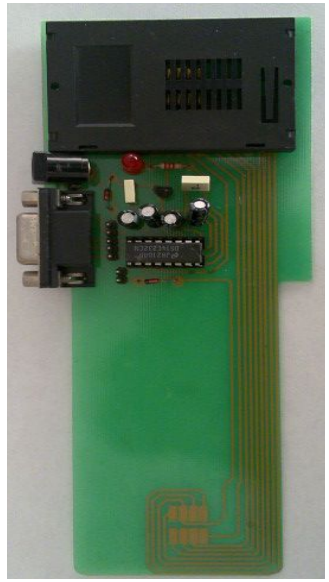


Tips : a 3,57MHZ clock is the norm. But overclocking the card at 6.00MHZ is fine, faster zapping !

SECA 2 : hardware for sharing

Client side : Modified rented receiver (JTAG) with extra code added to grab session KEY over a serial port + a PC or a Linux embedded device to run a session manager connected to internet.





SECA 2 : countermeasures

- An annoying feature : an extra encryption on the signaling datas. Usually AES is used and the signaling is decrypted by the receiver before sending ECM to the card. To solve this problem, needs to reverse the firmware of an official running set top box and extract the AES KEY needed to clean the ECM before send it to the card server.

- Until 2012, no real countermeasures on card sharing. Nowadays a rate limit on a card : no ECM serial flow from different channels are allowed. Even a fast zapping can lock the card and need a RESET to run again.

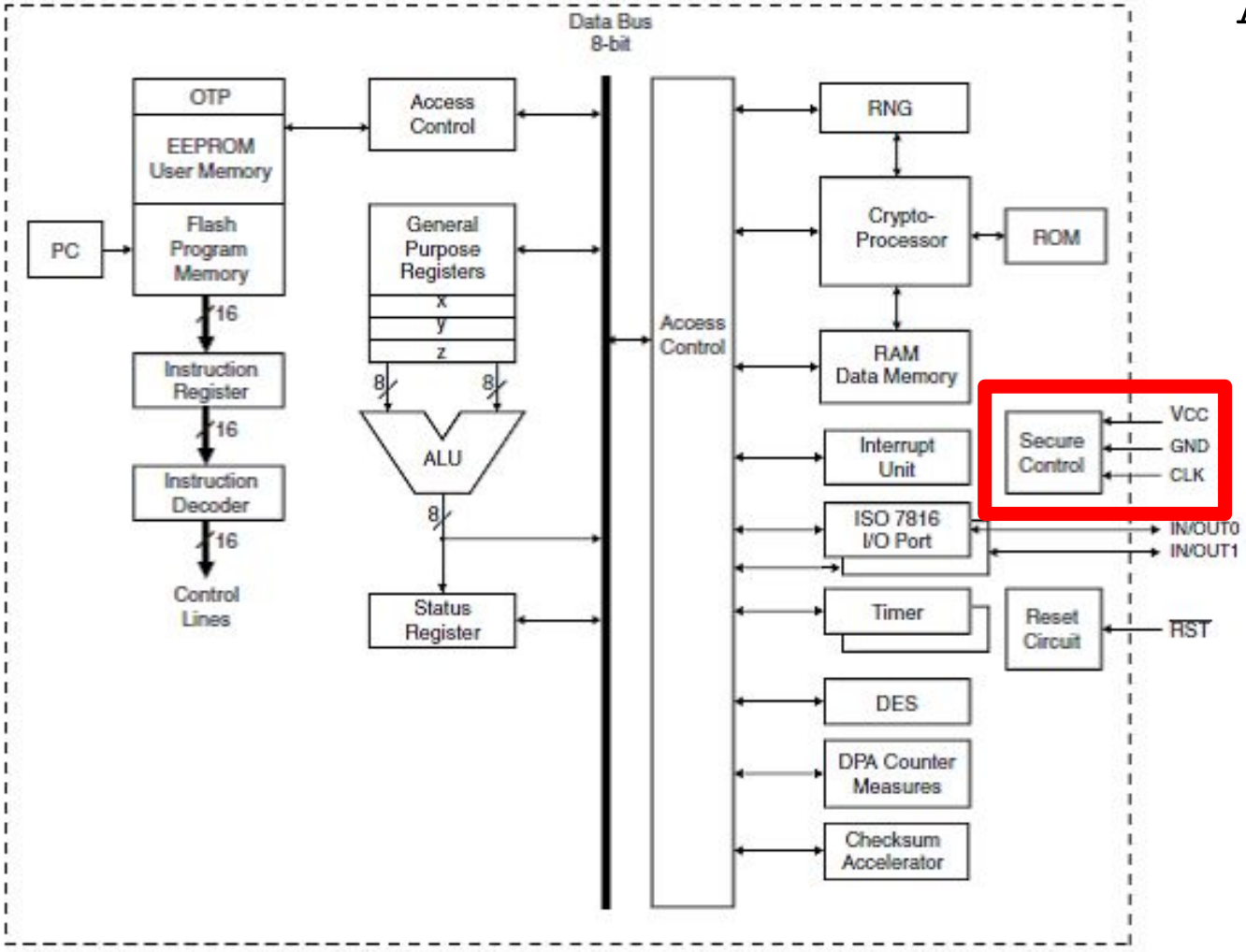
SECA 2 : The smartcards

3 ROM : but all the reverse based on the V7.0 from Italia, because the dump was available.

➔ The studying of the 64KB ROM show no strong software bug, only the ability to write datas/instructions in EEPROM, but no way to jump in. Useful but not enough ...

So, let's go for the real stuff :
Faults injection

SECA 2 : The microcontroller



ATMEL 90SC6464C

On the paper :
« Secure Control »

Black Hat 2008 say:
«The most secure of the available OTS choices».

Bad luck for us...

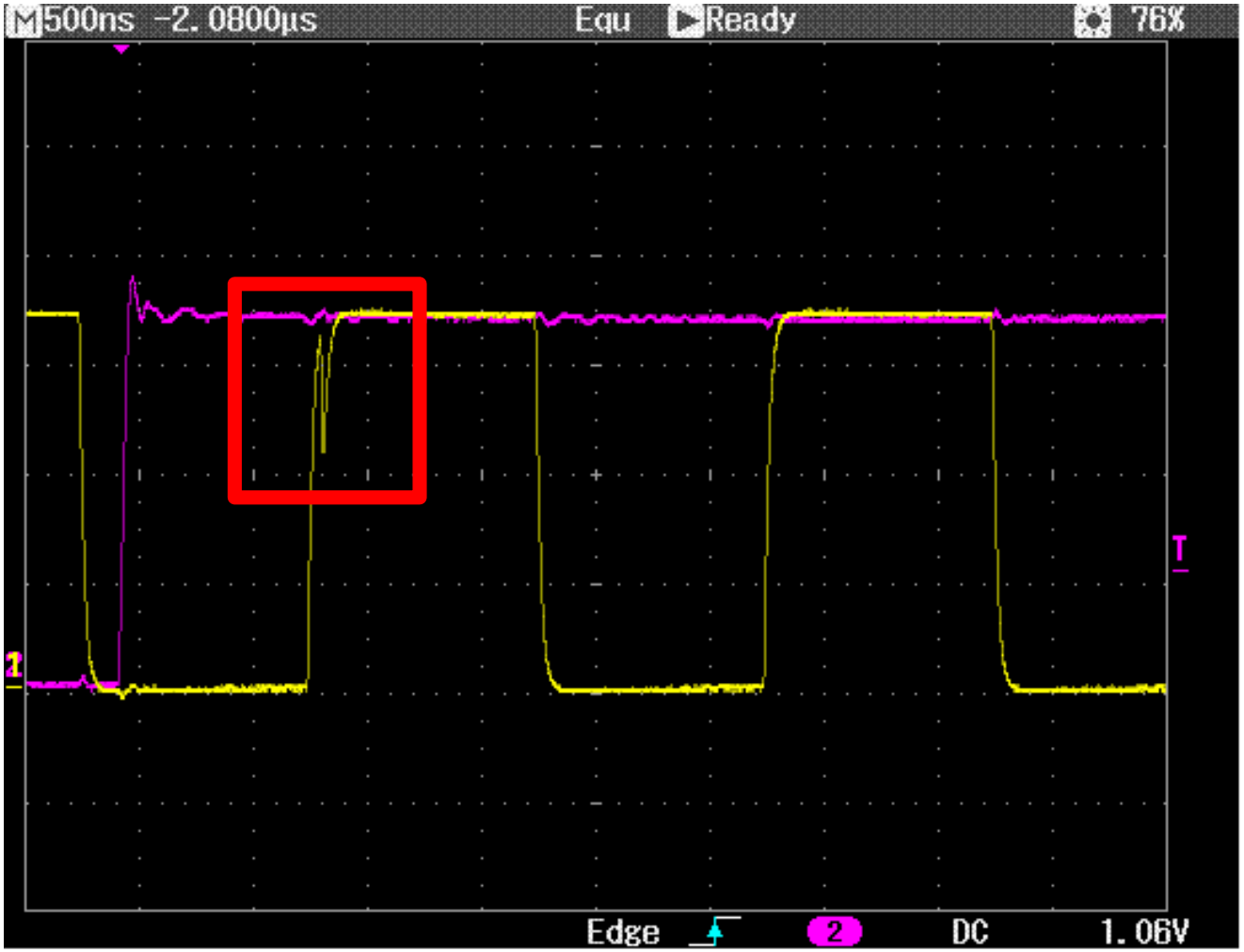
Ok, but let's check this feature

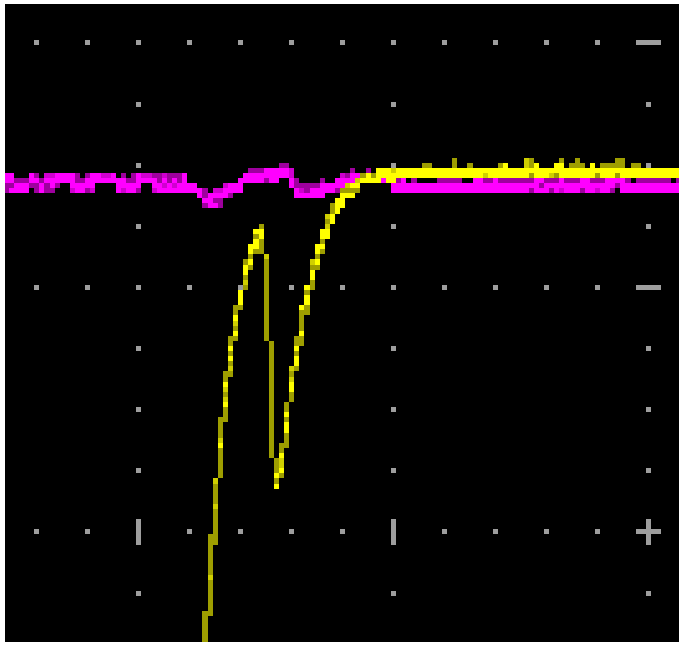
SECA 2 : Glitches

- At 3.57MHZ : trying glitches on VCC and CLK, nothing other than reset even or sometime kill the card.
- At 800KHZ : no way with VCC, but random faults obtained with the CLK.
- At 1MHZ : same behavior.

But we needs stability and reproduce the behavior

SECA 2 : THE glitch



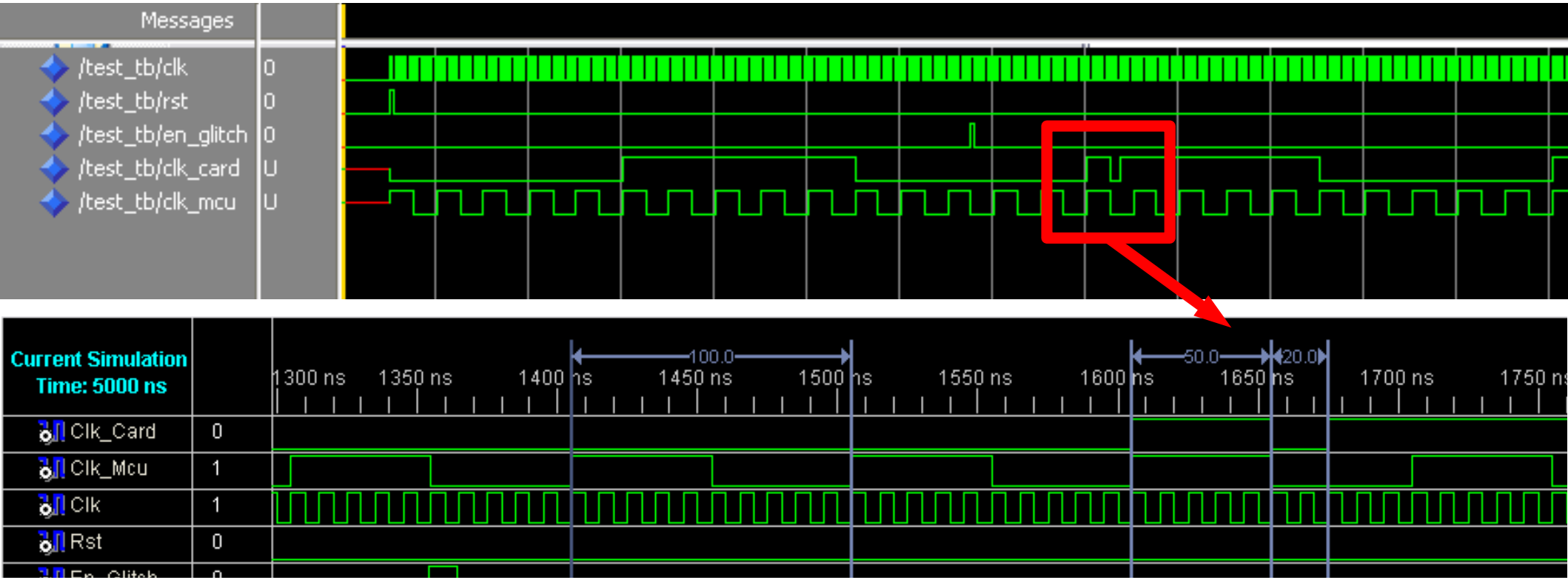


With CLK=1MHZ, glitch 50ns after the edge, pulse is 20ns wide. Works each time : skip an instruction.

SECA 2 : The CPLD

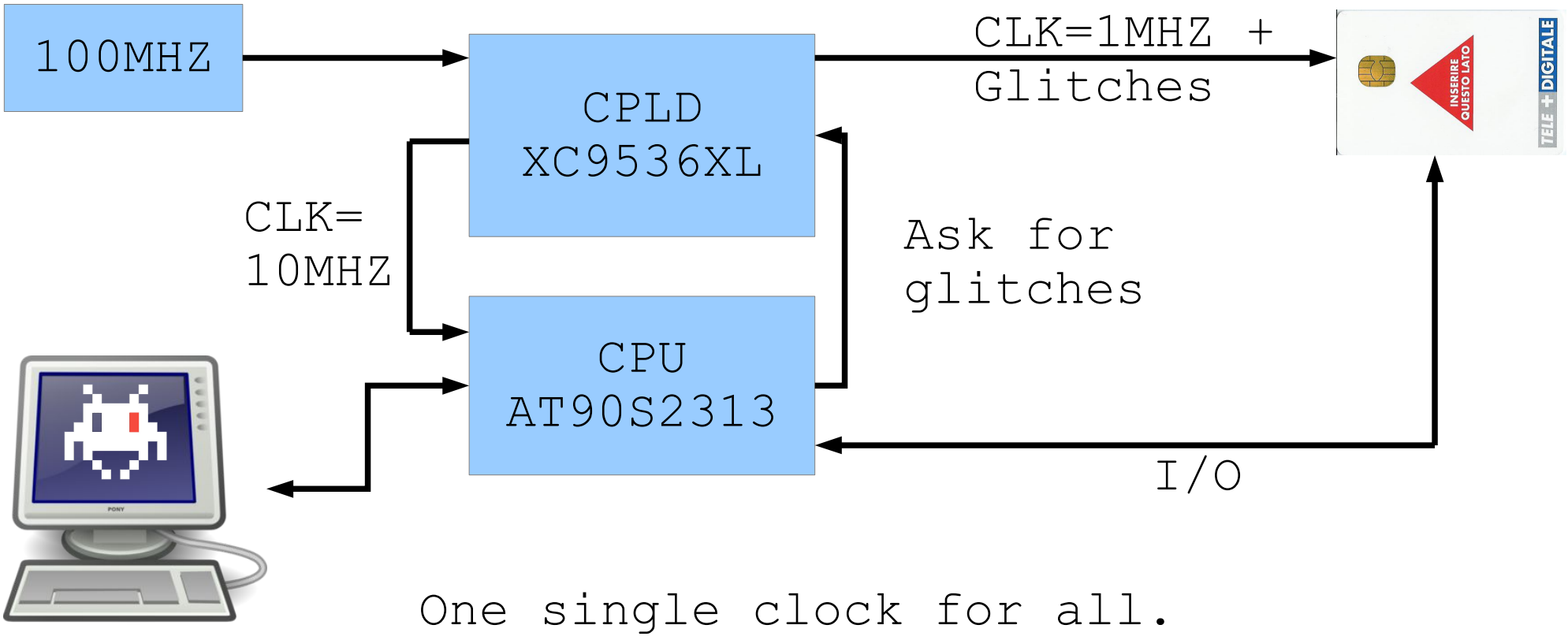
Simulation of the CPLD core program

Main function of the CPLD : the glitches generation



SECA 2 : The « unlooper »

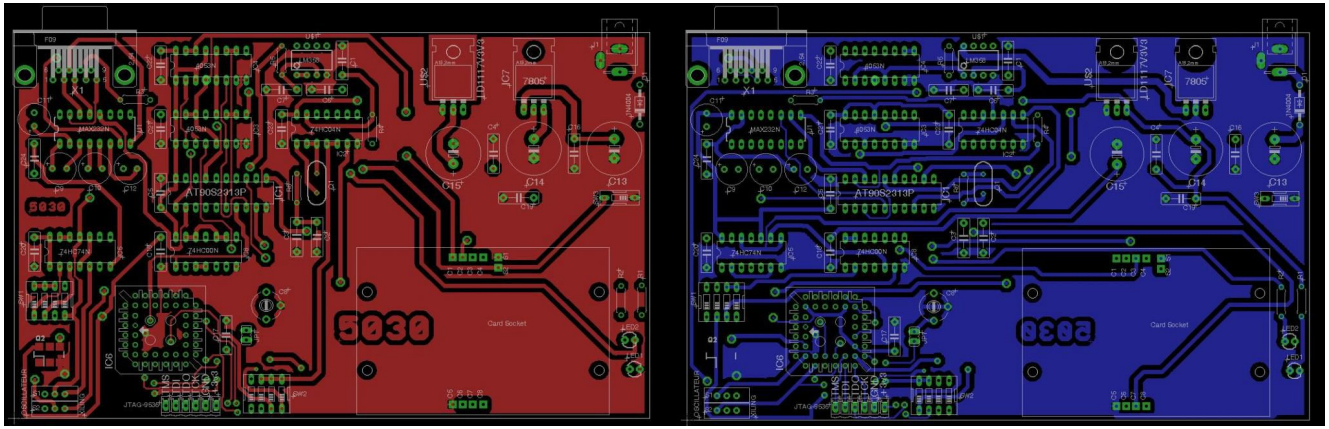
How it works :

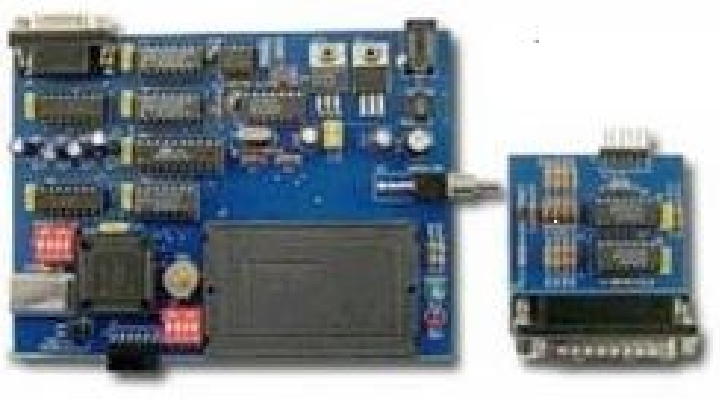


One single clock for all.
Card CLK / CPU CLK ratio : 1/10
Enough for accurate cycles counting

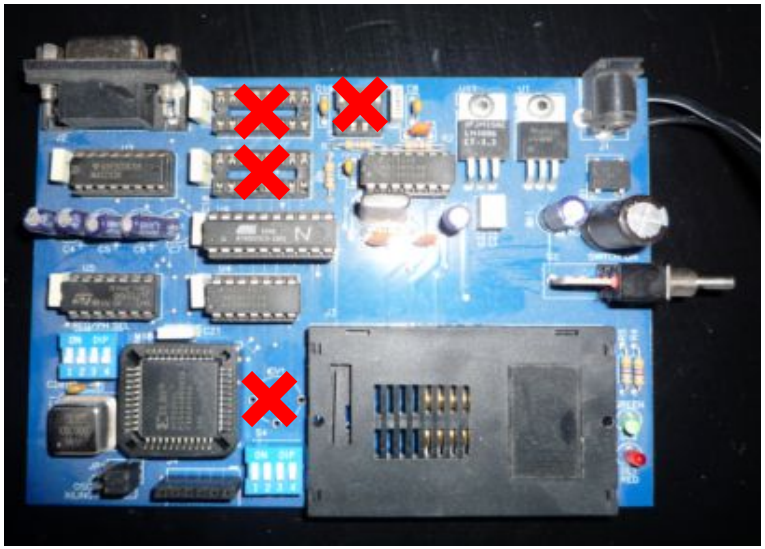
SECA 2 : The « unlooper »

50ns, 20ns, means a clock of 100MHZ.
Hard for hobbyists like us …
But this can be bought for a few bucks :





Cut & strap few connections, replace the CPLD, remove IC slowing lines, the modded one is ready and stable :



SECA 2 : Synchronise it

Synchronize the CPU & the card at the cycle.

- Need a reference point to start count cycles : when the last bit of the command is send, the CPU start to count.
- The ratio of 1/10 between card & CPU allow to be accurate and never miss a cycle.

So, we are ready to hit an instruction, but how to count cycles ?!?

SECA 2 : Count the cycle

The needs of a simulator …

Manually count cycles is impossible : glitch an instruction after 55709 cycles can't be counted on our finger …

«AVR studio» help us on this point.

But we needed to mod the instructions of the full ROM because AT90SC6464C is not supported.

SECA 2 : Random delay

Another security feature : random delays in execution flow. 100% of the code in the card is non constant timing.

Was a problem without the dump to reproduce behaviors; but with the dump, just need to skip the calls to the function.

SECA 2 : Stack manipulation

ROM dump show us a nice function : A command (C1 B4) doing RAM copy.

With the help of 5 glitches and special crafted parameters, this command can write in RAM any bytes.

This can be used to override the return address in the stack and jump in EEPROM.

SECA 2 : List of the tools

In our toolbox we have now :

1. Software bug to write 4x22 bytes in EEPROM.
2. Hardware (unlooper) to handle the card and hit any instruction in execution flow.
3. A AT90SC6464C simulator for ROM cycles count & running code.
4. Stack manipulation ability.

SECA 2 : THE attack !

Scenario for running our own code :

1. Assemble and write the code in EEPROM using the software bug.
2. Use unlooper to send the glitches while sending C1 B4 command to jump in the EEPROM.
3. Get the result of our code as a legacy command would return values.

SECA 2 : Applications

Benefits of running our code :

1. Dump every part of ROM/RAM/EEPROM for studying purpose, but also :
   - Monthly key
   - User management key
2. Replace any byte in EEPROM :
   - Change RSA private key to avoid counter measures.
   - Replace serial number, management key from a valid customer.

SECA 2 : Backdoor

Now let's write our backdoor :

Another nice feature is the ability for the provider to send upgraded code in the card. Using this feature allow us to run homebrew functions, but WITHOUT unlooper.

With a simple ROM / RAM / EEPROM management backdoor, you have the full access with a simple ISO 7816 reader.

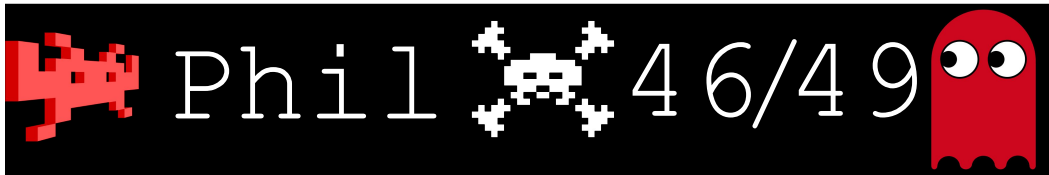
SECA 2 : Backdoor

```
Main:
                ldi     r16,0x55
                call    0x789D          ; Routine d'émission
                clr     r18
                out     RAMPZ, r18      ; RAM sélectionnée
                ldi     r31,0x9         ; Adresse MSB de destination
                ldi     r30,0x60        ; Adresse LSB de destination
                ldi     r17,0x85        ; 133 octets à recevoir

rGetIOByte:
                call    0x78DB          ; routine de réception sérielle
                st      Z+,r16
                dec     r17
                brne    rGetIOByte

                ldi     r16,0x55
                call    0x789D          ; acquitement

                ldi     r26, 0x60       ; adresse LSB de départ
                ldi     r27, 0x09       ; adresse MSB de départ
                ld      r18, X+         ; 0x96F --> type de support RAM/eeprom/rom
                ld      r30, X+         ; $x970 --> adresse LSB
                ld      r31, X+         ; 0x971 --> adrese  MSB
                ld      r25, X+         ; 0x972 --> len
                ld      r19,X+          ; 0x973 --> zone d'ecriture
                cpi     r19,0x40        ; DUMP EEPROM/RAM
                breq    InitSupport
                cpi     r19,0x80        ; DUMP ROM
                breq    RdFLASH


                movw    r17:r16,r31:r30
                st      -y,r19
                st      -y,r25              ; Len pour l'ecriture
                ldi     r20, 0x65
                ldi     r21, 9
                call    0x4505              ; routine d'écriture
                rjmp    Main

InitSupport:    out     RAMPZ, r18
RdEEPROM        ld      r16,Z
                call    0x789D
                adiw    r30, 1
                cp      r31,r25
                breq    RdFinish
                rjmp    RdEEPROM
RdFinish:       rjmp    Main

RdFLASH:        lpm     r16,Z
                call    0x789D
                adiw    r30, 1
                cp      r31,r25
                breq    RdFinish
                rjmp    RdFLASH
```

API :

133 bytes mandatory :

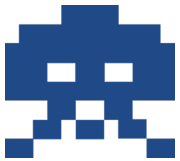
Byte1 = FLASH or RAM
Byte2:Byte3 = ADR
Byte4 = LEN
Byte5 = Write EEPROM or dump ROM or EEPROM
Byte6:Byte133 = datas



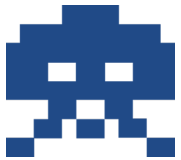
SECA 2 : countermeasures

Some try to hit cloned cards, but as we disallow the code upgrade ability (backdoor & RSA KEY), no way for the provider to gain again control !

With our degree of management of the card, no countermeasures were really possible …



We Onw U !



SECA 2 : Emulators

As every secret was available, implementation of emulators in cheap smartcards with onboard fast enough RSA / SHA1 was possible with full auto-update features.

This time a strong countermeasure appears : The card wasn't based on a standard AT90SC6464C … A modified version with a custom designed crypto-engine was used. Some basics functions were recovered by cryptanalysis (or unknown means), but the strong customs functions remind today undiscovered.

They Own Us !

This is the end ...

To conclude :

- Beware of hobbyist with no means, time, determination and … brains.
- Fun !
- The SECA 3 is not compromissed, give it a try ?

- Questions ?