

---

# Memory address scrambling revealed using fault attacks

Jacques Fournier

CEA-LETI

[jacques.fournier@cea.fr](mailto:jacques.fournier@cea.fr)

Philippe Loubet-Moundi

GEMALTO

[philippe.loubet-moundi@gemalto.com](mailto:philippe.loubet-moundi@gemalto.com)

# General Context

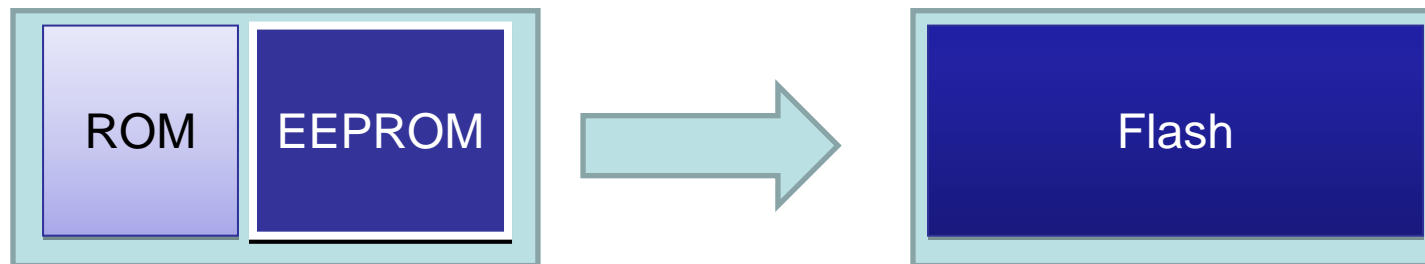
---

- Attacks on security devices:
  - Logical attacks
  - Invasive attacks
  - Side-channel attacks
  - Fault attacks... *semi-invasive attacks*
- Our focus is on fault-induced attacks to corrupt memory contents
- We target remanent faults induced on Floating Gate (FG) memories like EEPROM and Flash

# Motivations

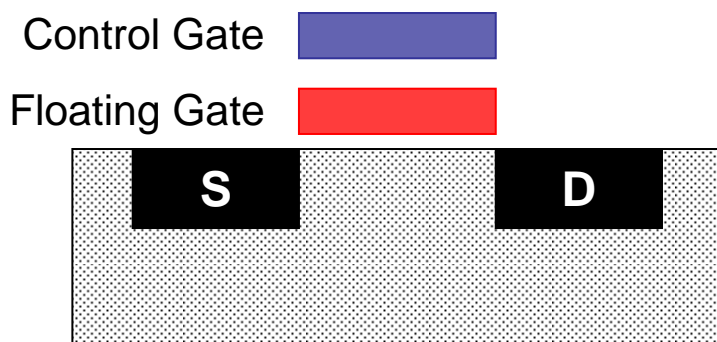
---

- Develop tests useful for current secure IC's evaluation
- No timing constraints (hackers can wait)
- Compare different suppliers scrambling techniques
- Anticipate risks associated to the evolution of the smart card market trend

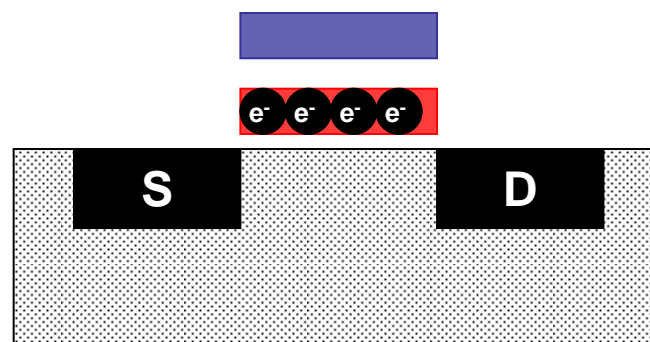


# FG memory cells

- The state of the FG cell is determined by the presence of electrons ( $e^-$ ) or not



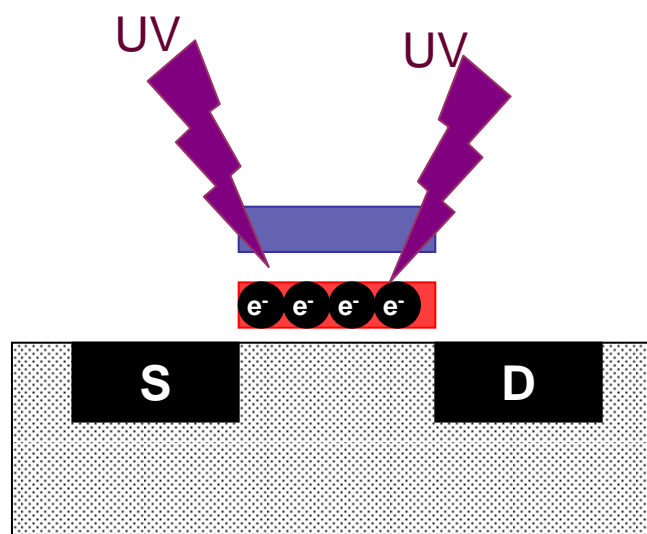
**'0' : no charge**  
*(erase state)*



**'1' : e<sup>-</sup> in the Floating Gate**  
*(programmed state)*

# Effects of UV light on FG cells

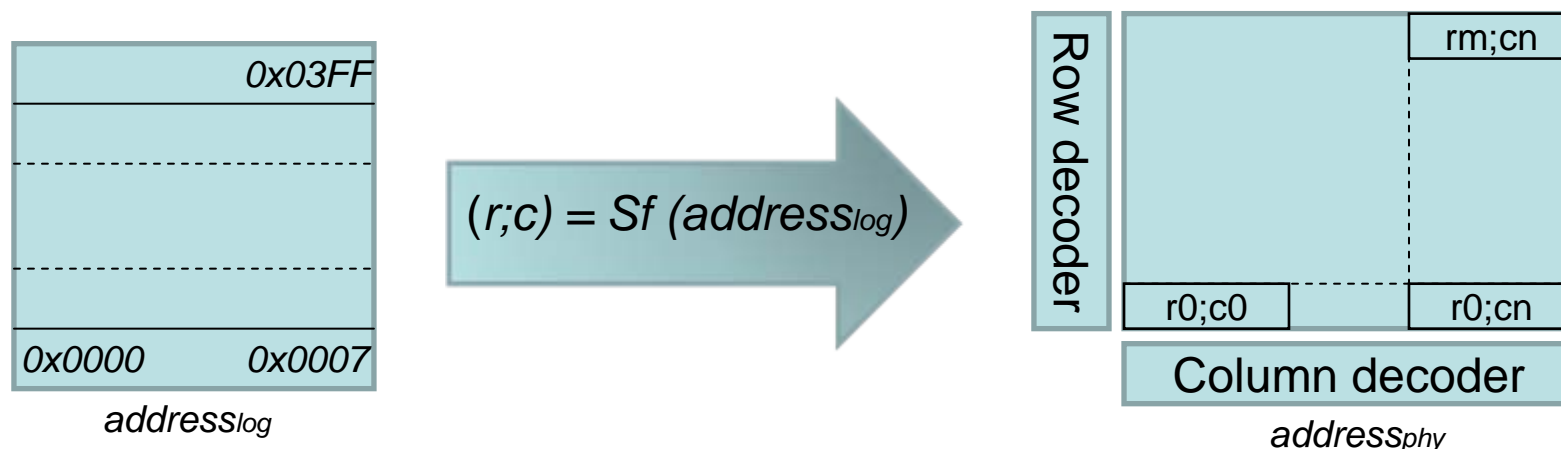
- UV light would 'knock off' electrons from the Floating Gate



- Usually accounted for by the use of passivation layers ( $\text{SiNx/SiO}_2$ ) and of a protective polyimide layer...

# Address Scrambling

- In most security devices, ‘address scrambling’ is used on FG memories
  - No direct mapping between the physical location of a given data and its logical address



- Acts as a counter-measure against an attacker willing to physically change the value of bit(s) to corrupt a sensitive piece of data whose logical address he might know

# Experiments' set-up

- UV source is 'standard' UV eraser delivering a power of approx. 4000uW/cm<sup>2</sup>



- DUT is a simple smart-card chip having 16KBytes of EEPROM and a 'native' OS interpreting simple APDU commands (memory program, read, erase...) in 0.35um technology
- The plastic encasing was scratched off and the epoxy resin layer was removed using fuming HNO<sub>3</sub>

# Experiments' scenarios

---

- Expt I: Qualification of EEPROM sensitivity to UV light
- Expt II: Reverse-engineering of memory address scrambling
- Expt III: Modification of specific bytes



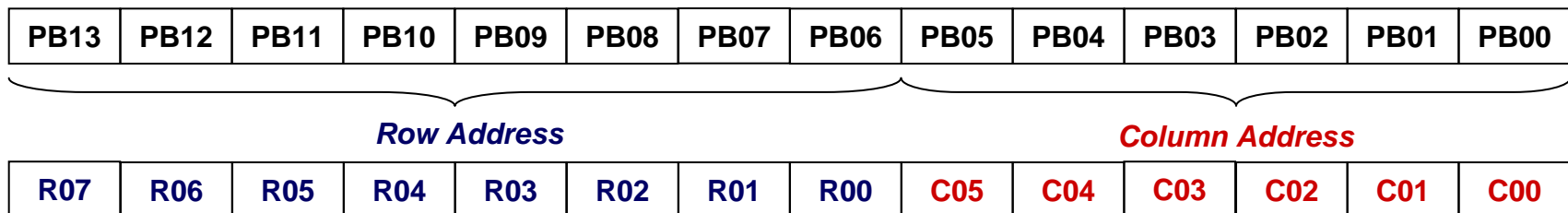
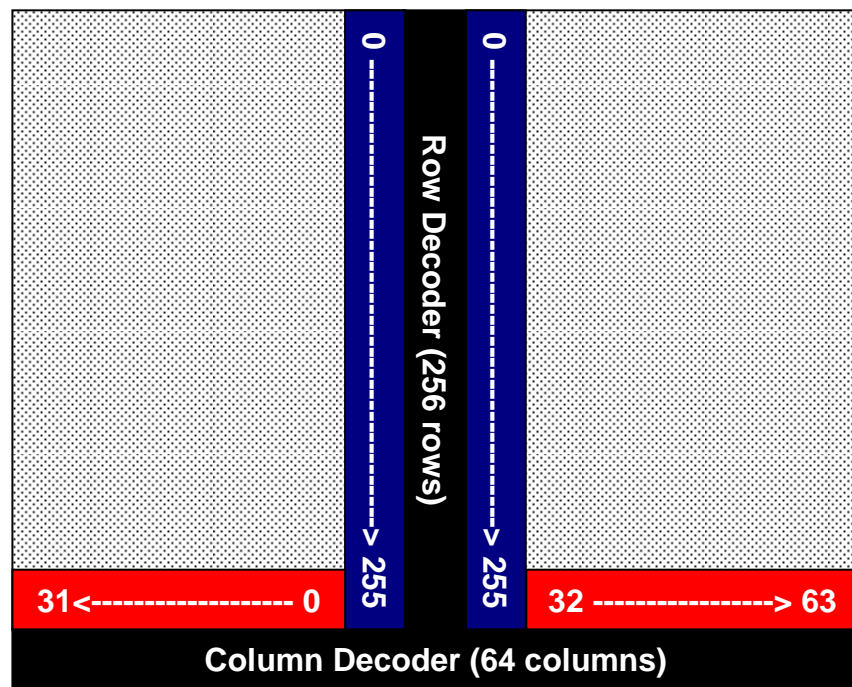
# Expt I: Qualification of EEPROM sensitivity to UV light (1/1)

---

- Scenario:
  - The memory was programmed with 1's.
  - The memory was read to check that all 1's are present.
  - The memory is exposed to UV light during which the memory is read regularly.
- Observations:
  - After 7 hours, 142 bytes had been 'erased'.
  - Some bytes' state flipped from one reading to another illustrating that the cells needed more UV exposure for a stable state.
  - A longer exposure definitely flipped the bytes to '0'.
- Conclusions:
  - Experiments confirm the sensitivity of EEPROM to UV light.
  - For the DUT's technology (0.35um) longer times of exposure are needed when compared to previous publications that reported experiments on larger technologies (0.9um).
  - *(No memory encryption: physical zeros where read as logical zeros).*

# Expt II: Reverse-engineering of memory address scrambling (1/6)

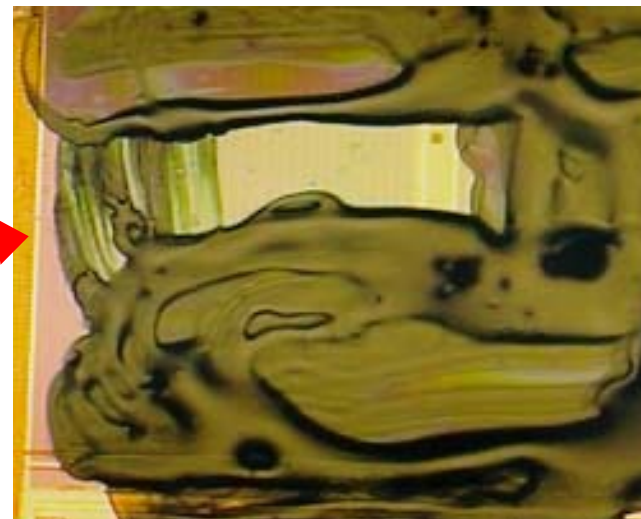
- Initial Observations:
  - Using a microscope, we physically observed the organisation of the memory arrays.
  - Two symmetric blocks where each has 32 *columns* by 256 *rows*
- Initial Conclusions:
  - 14 bits for addressing (16K memory).
  - The addresses are split into 8 bits for the row number and 6 bits for the column number.



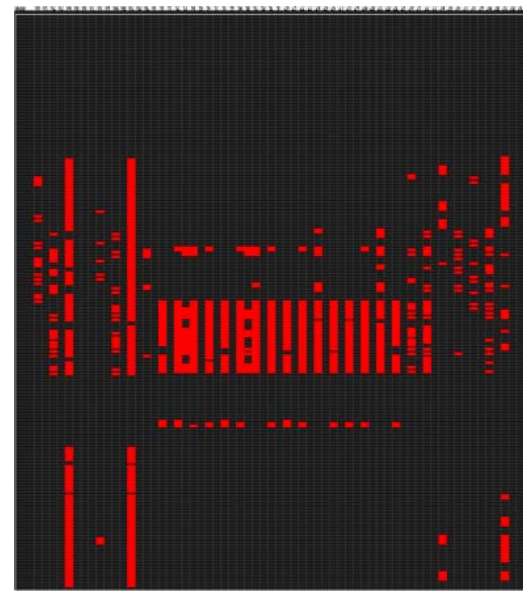
Physical Address

# Expt II: Reverse-engineering of memory address scrambling (2/6)

- Scenario:
  - The memory was programmed with 1's...
  - Part of the memory was hidden with an opaque cache using an overhead projector pen
  - The memory is exposed to UV light for 14 hours before reading back the memory again
  - A logical mapping of the memory (black for '1's and red for '0's) was drawn



- Observations:
  - The logical mapping did not correspond to the physical space not cached

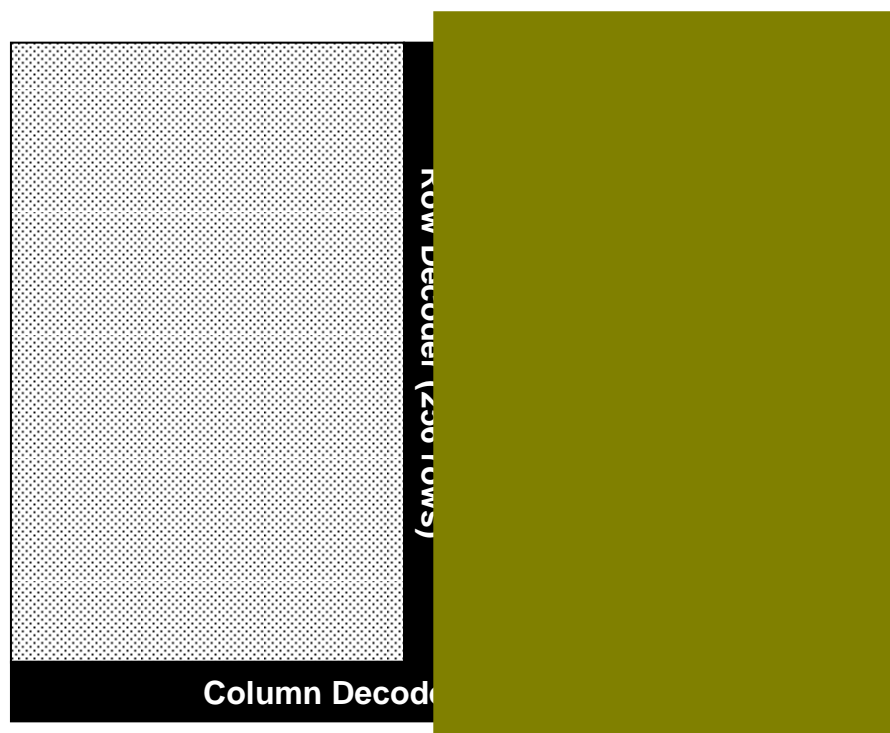


# Expt II: Reverse-engineering of memory address scrambling (3/6)

---

- Scenario:
  - We selectively ‘hid’ part of the physical memory so as to infer which bit of the physical address should be varying
  - We exposed the memory to UV light
  - We read the memory and built the logical memory map, which allowed us to find which bit of the logical address is modified
    - This allows us to have the mapping between a given physical address bit and a given logical address...
  - We started over again with another part of the memory... and using the preceding results

# Expt II: Reverse-engineering of memory address scrambling (4/6)



We hid half of the memory

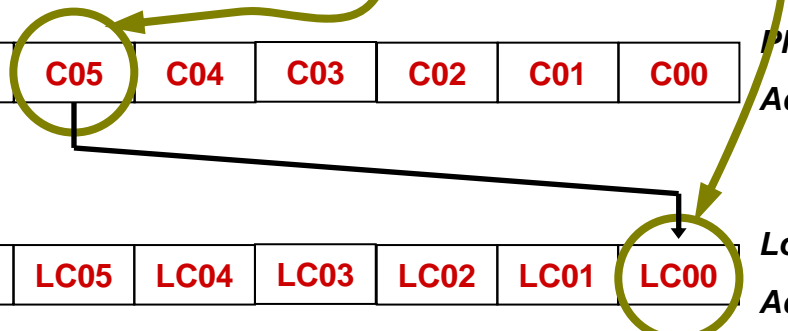
On logical mapping, we saw that one out of two columns was modified

R07	R06	R05	R04	R03	R02	R01	R00	C05	C04	C03	C02	C01	C00
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

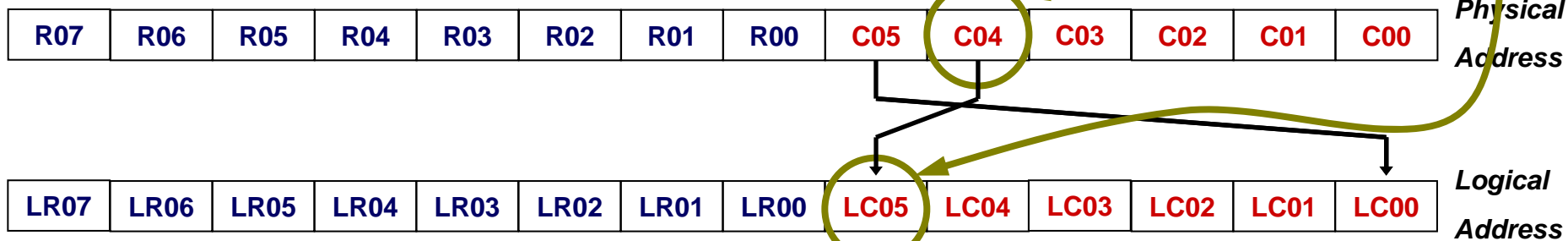
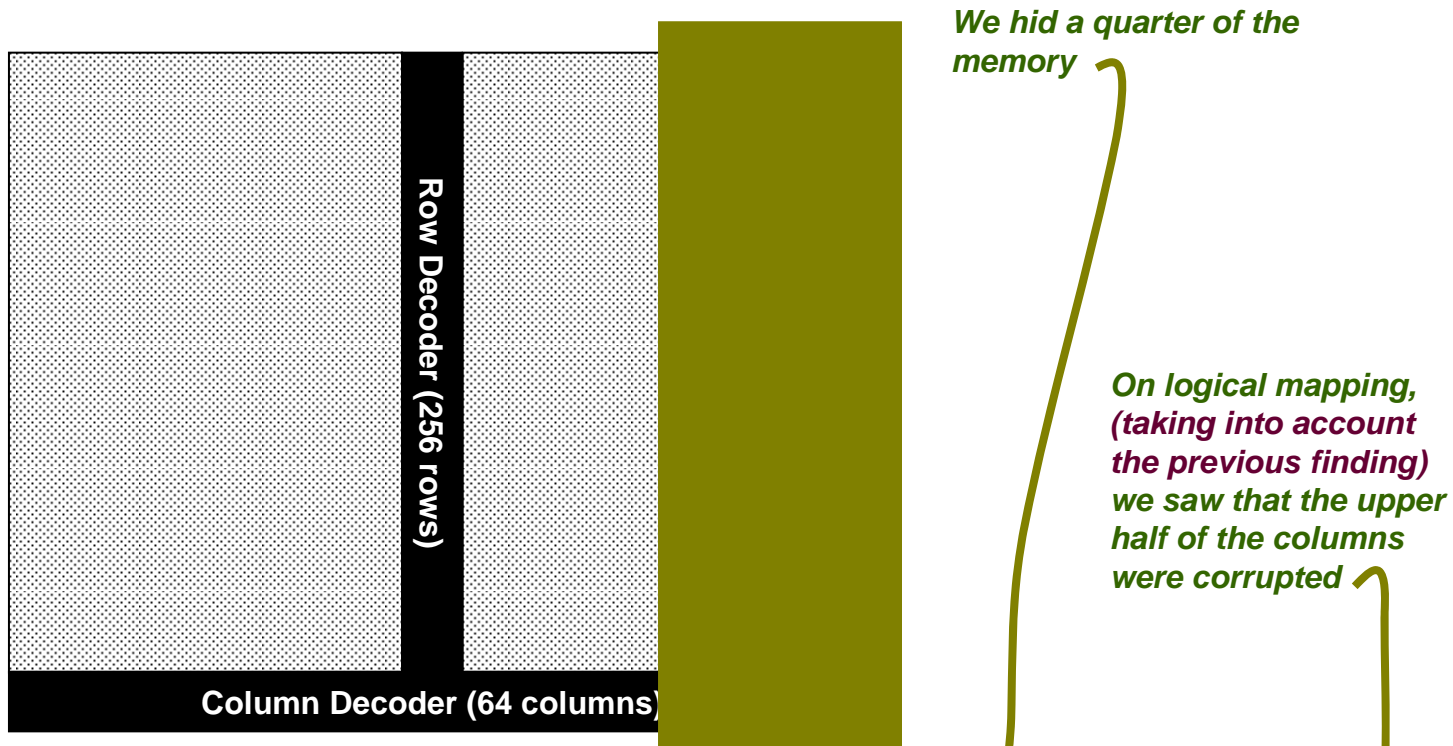
Physical Address

LR07	LR06	LR05	LR04	LR03	LR02	LR01	LR00	LC05	LC04	LC03	LC02	LC01	LC00
------	------	------	------	------	------	------	------	------	------	------	------	------	------

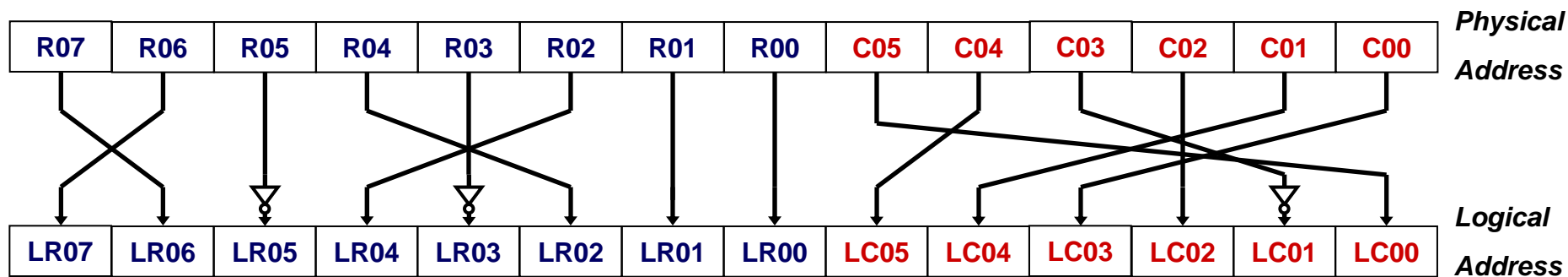
Logical Address



# Expt II: Reverse-engineering of memory address scrambling (5/6)

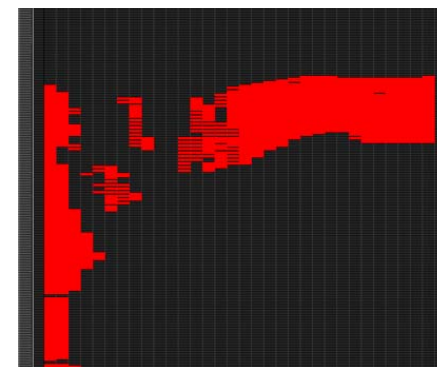


# Expt II: Reverse-engineering of memory address scrambling (6/6)



*Our first experiment...*

*Logical Mapping*



*Physical Mapping recovered once the scrambling formula has been applied*

# Expt III: Modification of specific bytes (1/2)

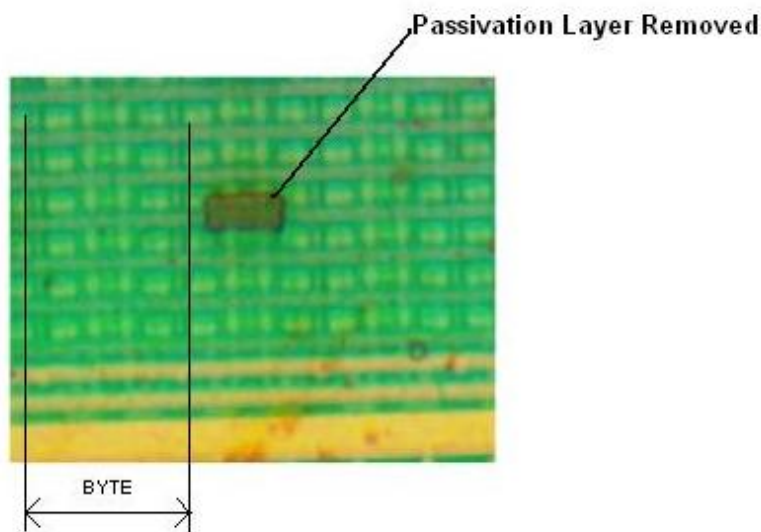
---

- Scenario:
  - As the scrambling is known, we wanted to test what is the minimum number of bits that we can modify on a targeted byte
  - We remove the passivation with the minimum possible window aperture given by our laser
  - We exposed the memory to UV light
  - We read the memory and analyze the content of the targeted byte (how many bits have changed)



# Expt III: Modification of specific bytes (2/2)

---



- 4 bits were modified
- Adjacent rows could have modification
- Scattering effect of UV exposure



Better set up needed to modify only one bit

# Exploitations

---

- Modify sensitive data (fuses, security flags...) whose logical addresses are known
- Modify opcodes to corrupt code execution
- Technique to test the resistance of a particular device's EEPROM to UV erasure
- *Simple & low cost technique*

# Limitations

---

- We can only convert 1's to 0's (programmed to erased state)
- No metal shield on our DUT, few metal layers
- Might be more tedious with smaller technologies
  - Scattering effect would be more important
  - Advanced techniques or expensive tools (FIB) might be used
- Today's use of ECC on EEPROMs might hinder such attacks
- Counter-measures have already been deployed on latest smart cards

# Counter-measures

---

- Logical counter-measures
  - Sensitive data have redundant complementary representations in memory
  - Integrity checks on sensitive data read from memory
- Hardware counter-measures
  - Use of a top metal layer or a metal grid
  - More complexe scrambling algorithm
  - Scrambling which might vary from one die to another
  - Use of error correcting codes
  - Use memory encryption

# Conclusion

---

- We developed techniques that improve UV tests on secure IC's
- We succeeded in retrieving the scrambling of NVM memories, using this procedure
- We evaluated the feasibility of specific bit modifications using UV exposure
- We proposed several counter-measures to make those experiments much more complicated

---

**THANK YOU FOR YOUR  
ATTENTION**



