How microprobing can attack encrypted memory

Sergei Skorobogatov

http://www.cl.cam.ac.uk/~sps32

email: sps32@cam.ac.uk

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

Introduction

- Hardware Security research since 1995
  - testing microcontrollers and smartcards for security
  - research on semi-invasive attacks (PhD, 2005, Cambridge UK)
  - backdoors in semiconductors (2012)
  - iPhone 5C NAND mirroring (2016)
  - “impossible” solutions for challenges in real-world devices
- Hardware Security is about finding flaws and fixing them
  - preventing attacks on service, data and IP theft
  - what security features are implemented and how secure they are
- Hardware Security challenges
  - data, keys and passwords storage
  - new attack technologies
  - modern fabrication processes (10nm, 14nm, 28nm, 45nm, 65nm)
  - developing countermeasures through understanding of flaws
  - prediction of new attack methods

Introduction

- Secure devices
  - smartcards and microcontrollers
  - FPGAs and ASICs
  - secure memory
- Embedded memory
  - Mask ROM: bootloader, firmware, algorithms
  - EEPROM: variables, keys, passwords
  - Flash: bootloader, firmware, algorithms, keys, passwords
- Memory extraction is the crucial step in attacks
  - access to firmware for reverse engineering
  - extraction of crucial algorithms
  - access to sensitive data, keys and passwords

Secure memory extraction methods

- Mask ROM
  - deprocessing and imaging
  - deprocessing, staining and imaging
  - scanning probe microscopy (SPM): slow – hours per kB of data
  - microprobing
- Flash and EEPROM
  - using standard interface after defeating the security (JTAG, ISP...)
  - microprobing (undocumented and custom devices)
  - scanning probe microscopy (SPM): slow – hours per kB of data
  - scanning electron microscopy (SEM): limited – down to 130–180nm
- What if the memory is scrambled or encrypted?
  - reverse engineering of the decryption logic (slow)
  - microprobing inside the CPU bus (reverse engineering of logic)
  - software attacks on bootloader/firmware flaws
  - brute forcing the scrambling/encryption

Invasive attacks

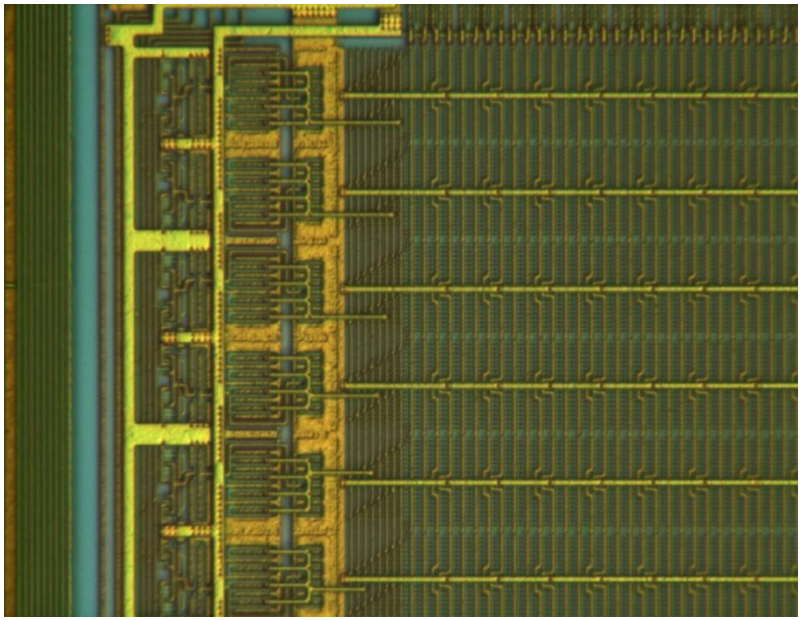
- Mask ROM is challenging to attack
  - high density of cells
  - impossible to load with test pattern for cryptanalysis
  - could be ion-implanted for doped encoding
- How strong is the data bus encryption for low-end CPU?
  - 8-bit bus: $2^8! \approx 8.58 \cdot 10^{506}$
  - impossible to brute force
- How strong is the address bus encryption for 32kB ROM?
  - 15-bit bus: $2^{15}! \approx 9.09 \cdot 10^{133733}$
  - impossible to brute force

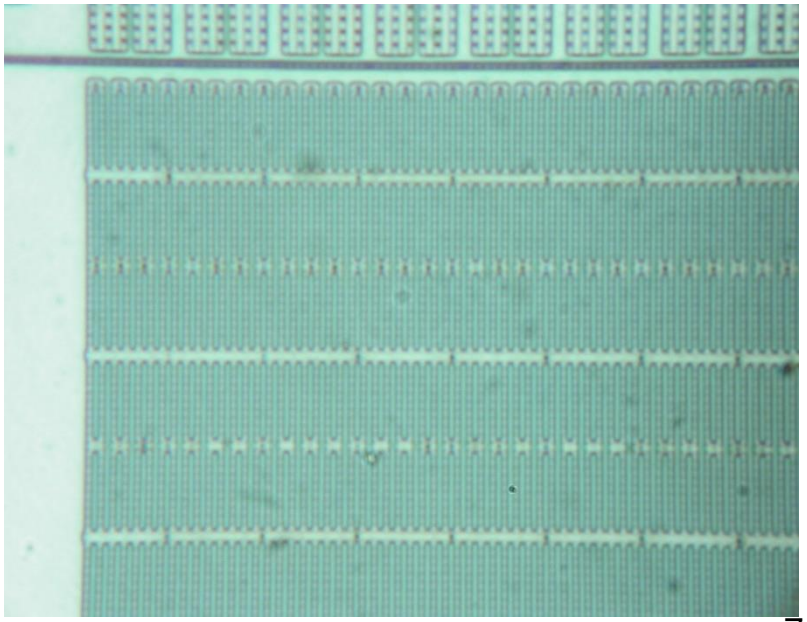
Attack implementation

- CPU executes code from Mask ROM and operates with plaintext data
  - observe the activity on the data bus (encrypted data)
  - inject arbitrary data into the data bus
  - observe response on both data and address buses
- Defeating data bus encryption
  - inject arbitrary data and observe response from the CPU
- Defeating address bus encryption
  - 90% of commands do not affect the program counter (PC), hence, sequential real addresses

Challenging target

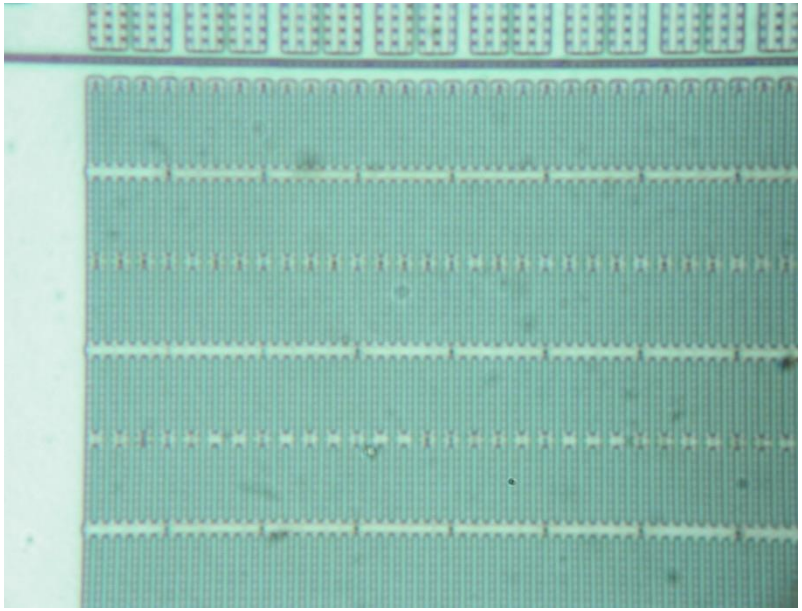
- Undocumented secure microcontroller used in car security
  - NEC78K/0 CPU core with encrypted Mask ROM, 0.35µm process
- Mask ROM deprocessing does not reveal any data – only structure of the transistors
  - 0s and 1s are encoded with different doping concentration of cells transistors – not visible under optical microscope

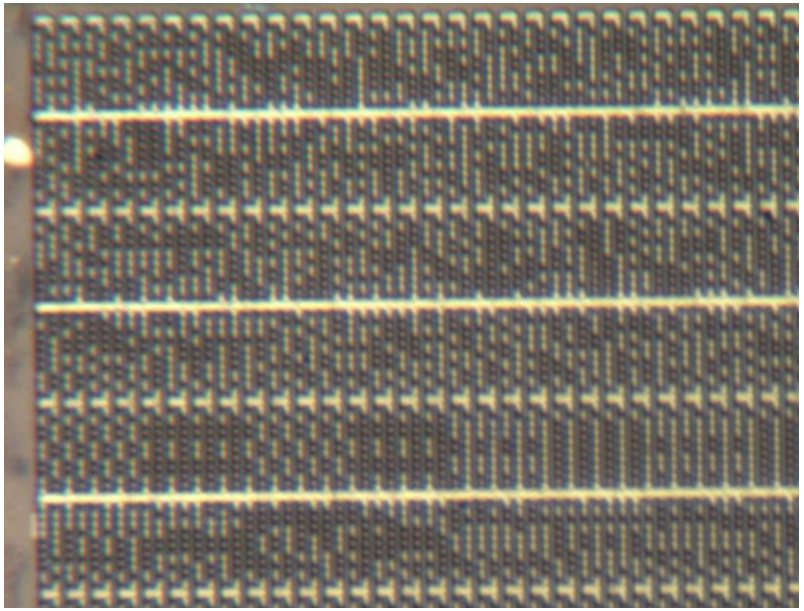




Data extraction

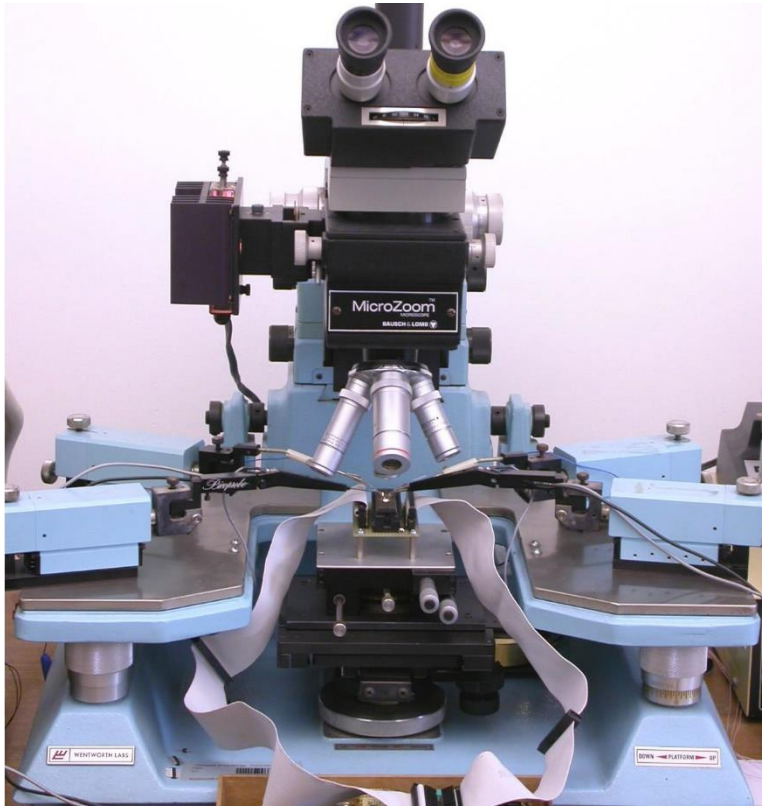
- Mask ROM deprocessing followed by scanning capacitance microscopy (SCM)
  - slow: hours per kB of data
- Mask ROM deprocessing followed by selective etching
  - chemical etching of silicon with speed of reaction dependent on the doping type and concentration: different trench depth for 0s and 1s





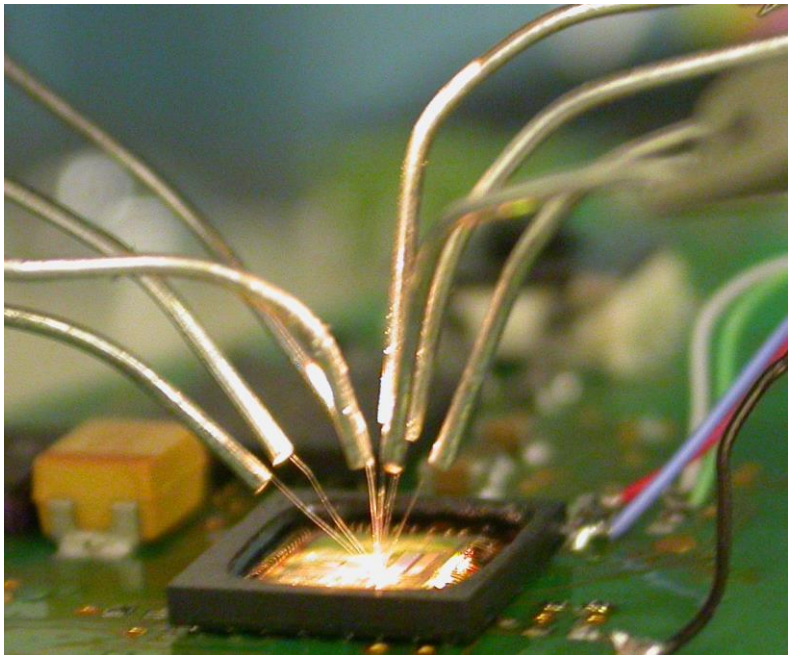
Experimental setup

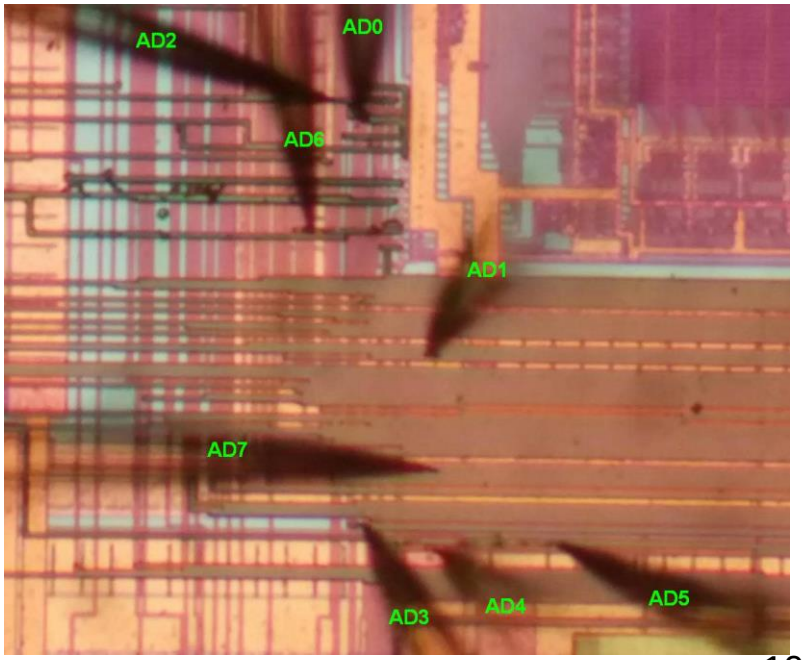
- Microprobing station for signal injection into the data bus
  - from $1,000 second hand on Ebay plus from $200 per probe
- Test board to control the device and the data bus
  - PIC24HJ128 microcontroller to drive RST, CLK and data bus



Experimental setup

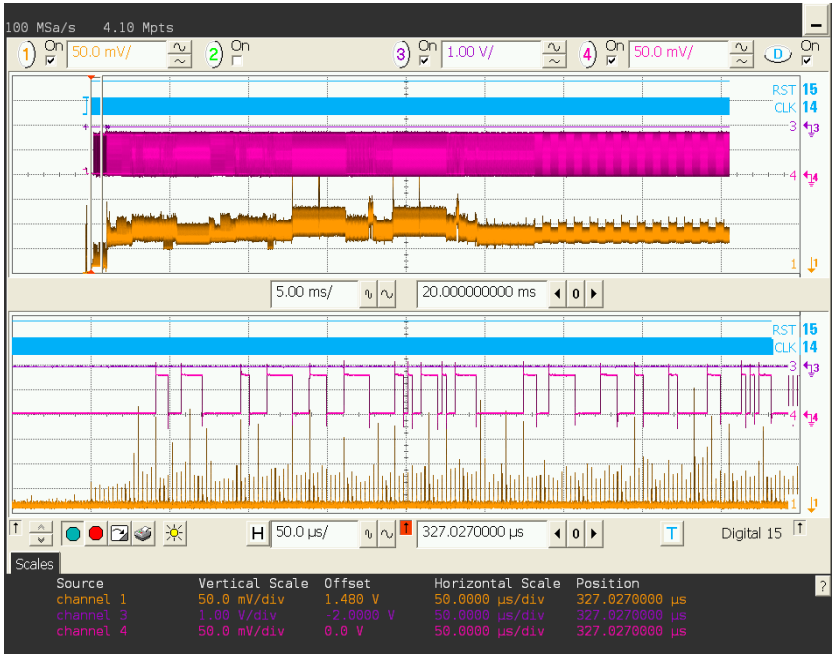
- Passivation layer was removed with laser cutter
  - from $3,000 on Ebay or use chemical methods
- GGB Picoprobe needles to establish contact with data bus
- Power analysis setup with 10Ω resistor for synchronisation
  - CPU runs from internal clock





Results

- Power trace immediately after Reset
  - internal unstable clock from on-chip RC oscillator
- Power trace 1ms after Reset
  - external clock, but hard to establish the moment of switching



Results

- First fetch from the data bus is the Reset vector (low 8-bit)
  - by injecting all values from 00h to FFh the correspondence table for address 0000h was built
  - for strong encryption that would be S-Box, but for this chip it was a simple scrambling such as XOR with constant
- Mind the address bus encryption
  - the first address is not 0000h because of the encrypted address bus
  - we have to let the CPU to run random code to build correspondence table for the address bus (90% of instructions do not affect PC)
  - the address bus encryption turned out to be XOR with constant 1Ch
- Now the correct correspondence table can be built
  - for the data bus at address 0000h it was a simple XOR with constant B6h

Results

- Second fetch is the high 8-bit of the Reset vector
  - in order to avoid microprobing of the full address bus we have to find the encrypted value that corresponds to 00h plaintext
  - that way we will force the CPU to start code execution from the address 0000h
  - that value turned out to be 3Dh
- The next fetch is instruction – we make it a branch
  - we force the CPU to execute **BR !addr16** (9Bh, low, high)
  - actual data injected into the data bus are: 2Dh, test, 00h
  - the second byte of the branch instruction is fetched from 0001h and is low 8-bit of the address
  - testing from 00h to FFh allows us building the correspondence table
  - for this chip it was a simple XOR scrambling with constant 3Dh

Results

- The process was repeated to fetch NOP followed by branch instruction
  - the correspondence table for address 0002h was built
- Finally we need to find the encrypted value for addresses 0003h/0004h that would be decrypted as **BR AX** command (31h, 98h)
- Then we can execute a simple program
  - **MOV A,!addr16**; load A with memory address specified in 0001/0002
  - **BR AX**; place plain text value of the memory on the address bus
  - appears on the data bus because of multiplexed A0-A7 with D0-D7
  - Reset CPU and repeat the process
- By executing the code for each memory address the contents of the whole memory space (64kB) can be extracted in less than an hour time

Implications and improvements

- Speed up the process by loading code into RAM and executing it
- The results were achieved on relatively old chip (0.35µm)
  - still serves well as proof-of-concept
  - demonstrates vulnerabilities of the CPU design that allow the secret information to be leaked on encrypted bus
- Microprobing of chips fabricated with 0.25µm and smaller process will require focused ion beam (FIB) machine
  - available from many universities at a cost of $50 – $100 per hour
- 16-bit CPU will require at least 16 probing points
  - more expensive probing station or custom test pad
- 32-bit CPU will require 2^{32} correspondence table
  - time consuming process – 5 days with 100µs per cycle time

Countermeasures

- Randomly delayed execution after Reset
- Dummy CPU cycles
- Limiting the number of Hardware Resets before Power-on Reset
- Prevent the CPU from executing an arbitrary code
- Use wider bus for embedded memory – it is impossible to build correspondence table for 64-bit bus

Conclusion

- Even the strongest encryption does not guarantee ultimate data protection because the information can be leaked by the party that has access to unencrypted data (e.g. CPU)
- Smaller fabrication process does not help on its own due to availability of sophisticated tools (e.g. FIB machine)
- Wider CPU data bus will make the attack harder with large number of test points and infeasible correspondence table
- The CPU must be designed with the hardware security in mind to prevent it from leaking sensitive information