Inducing Momentary Faults Within Secure Smartcards / Microcontrollers

Black Hat – Las Vegas

Christopher Tarnovsky
Flylogic Engineering, LLC.
chris@flylogic.net – http://www.flylogic.net
What is a momentary fault?

• Temporary forced change in behavior
• The change is precisely calculated
• A fault typically lasts no more than a few clock cycles
• Many faults may be executed to force favorable behavior other than normal
How do we do this?

- Low-capacitance buffered driver
- Driver is capable of driving a ‘1’ or ‘0’
- Driver is capable of listening in “Hi-Z”
- Low-voltage tolerant is a plus
Why would we do this?

• A series of changes can allow us too:
  • Overwrite stack-pointer
  • Force repeated loops in a code segment
  • Falsify cryptograms
Inducing the fault

- Physical connection to substrate
- Use low-capacitance buffered driver
- Tri-stated buffer is desired-
  - Allow eavesdropping
  - Overdrive at calculated point(s) in time
Eavesdropping (listening)

- **YELLOW:** Databus signal
- **PURPLE:** Reset
- **GREEN:** Clock
- **BLUE:** Trigger
Overdriving last slides databus with a logic ‘0’

YELLOW: Databus signal
PURPLE: Reset
GREEN: Clock
BLUE: Trigger
Initial steps

- Behavior analysis
- Understand your objective (the goal)
- Determine possible areas of attack
- Areas such as:
  - Address bus
  - Data bus
  - Cryptographic blocks
Address bus faults

- Likely choice for cryptographic memories
- Unlikely choice for microcontroller type devices
- Allows complete change of data bus
Data bus faults

- Most probable choice of attack
- Allows behavioral changes to many areas
Cryptographic block faults

- Limited use typically to Cryptographic Memory type devices
- Can allow readout of write only keys
Execution steps

- Determine-
  - When to induce the fault
  - How long to induce the fault
  - Do we need more than one fault
  - Can we execute more faults on the same line

- Execute the change of state during the period of time and see if the desired result occurs.
- If not, possibly take a “running log” of the bus
In Conclusion

• Most documentation seen tends to exaggerate the security level physically implemented
• Encrypted buses are just as vulnerable as non-encrypted buses
• Randomizing internal clock just means add a second needle
• Random software delays are unreliable

• Technology is improving but is not perfect
• Every standard secure IC made to date has been successfully compromised by hackers
• What is made by human can be taken apart by human