

On the technical limits of robustness rules

Cryptography Research, Inc.
www.cryptography.com

575 Market St., 21st Floor, San Francisco, CA 94105

© 2001-2005 Cryptography Research, Inc. Information in this presentation may be protected under issued and/or pending US and/or international patents of CRI. All trademarks are the property of their respective owners. The information contained in this presentation is provided without any guarantee or warranty whatsoever.



About Cryptography Research, Inc.

- 10 years of experience fielding security & anti-piracy systems
- Examples of technology designed by Cryptography Research:
 - SSL v3.0 / TLS v1.0
 - Most widely used security protocol in the world
 - In every major Internet browser
 - DPA countermeasures
 - Most smart cards made today use CRI's DPA countermeasure technology
 - Technology is in use in over 1 billion cards worldwide
 - CryptoFirewall
 - Secures pay television content in millions of set-top boxes
 - Self-Protecting Digital Content (SPDC)
 - Renewable security for next-generation optical formats



Building a speed bump...

- Suppose our goal is to prevent easy-to-repeat attacks
We might try a rule like:

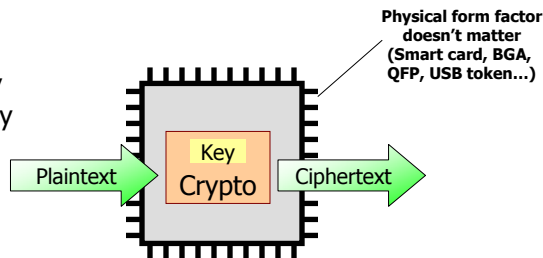
"Keys must be stored in the ASIC that does decryption, and it must be impossible to extract the keys without opening the chip package."

- On-chip key storage is achievable (though not always cheap)
 - Regularly done for smart cards & high-end security chips
 - ASIC fabrication processes are available that support NVRAM
 - "Only" a question of engineering effort, money, etc.
- But what about the security goal?
 - While compliance is inherently unverifiable, it sounds easy enough...



Hardware is secure, right?

Attacker goal:
Extract the key
cheaply & easily

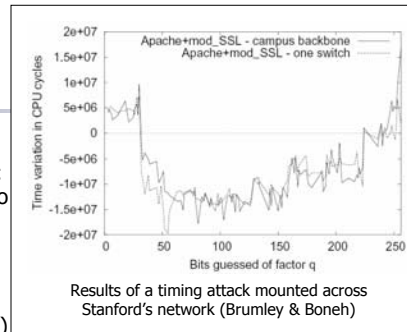


- This talk will look at a few examples of the attacks we have solve
 - To keep it simple, we'll ignore any attack that:
 - Requires technical skill to repeat (though attacks may take skill to devise)
 - Requires expensive or unusual equipment (budget: a PC plus \$0-\$500)
 - Costs any money to repeat
 - Example: Invasive attacks are excluded (decap costs ~\$100/chip)



Timing attack cryptanalysis

- The attack:
 - Measure the duration of cryptographic computations and analyze variations to find out the keys
- Risk: High
 - Developing attack software requires skill, but repeating the attack is trivial (e.g., software-only or a custom cable)
 - Can be mounted remotely
 - See: "Remote Timing Attacks are Practical" (<http://crypto.stanford.edu/~dabo/papers/ssl-timing.pdf>)
 - Differences of just one clock cycle (or <1 ns) are easily measured if the attacker has possession of the target device
- Countermeasures: Difficult
 - Make all operations take exactly the same amount of time
 - Many subtleties (Error cases, variable-time CPU instructions, cache hit/miss timing...)
 - Specialized cryptographic techniques make some variations non-exploitable



Glitching (fault induction)

- The attack:
 - Push the chip out of its valid operating conditions until it makes processing errors
 - Use these errors to find the cryptographic keys
 - This may sound difficult, but virtually any error (including random errors) can make it easy to break AES, RSA, and other cryptographic algorithms

Example: Breaking RSA

- RSA implementations spend >95% of their time on subexponentiation, so this is the easiest target:
 - $m_p = (c \bmod p)^{d_p} \bmod p$
 - $m_q = (c \bmod q)^{d_q} \bmod q$
- What happens if one of these operations is glitched?
 - Final result is correct mod p but wrong mod q (or vice versa)
- Given a correct answer m and glitched answer m':
 - $\text{GCD}(n, m - m') = p$ (or q)
 - The Euclidian algorithm quickly finds the private key

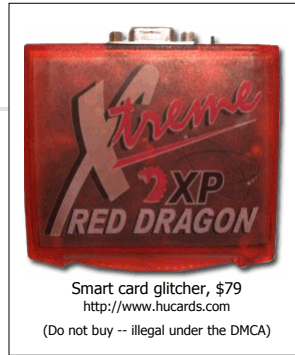


Power trace of a smart card being glitched during an RSA operation



Glitching (fault induction)

- The attack:
 - Push the chip out of its valid operating conditions until it makes processing errors
 - Use these errors to find the cryptographic keys
 - This may sound difficult, but virtually any error (including random errors) can make it easy to break AES, RSA, and other cryptographic algorithms
- Risk: High
 - Cheap and easy to repeat
 - Attack widely used by pirates & criminal hackers
- Countermeasures are very challenging
 - Board-level defenses are expensive (+ do not meet the requirement)
 - Analog filters, epoxy potting, ...
 - Main chip-level defense: analog circuitry to detect improper operating conditions
 - Tricky to avoid false-positives or false negatives (often requires 3-5 design iterations)
 - Limit chip pins – technology required to defend large chips with many power/clock inputs is not commercially available
 - Defenses are expensive & challenging to build successfully

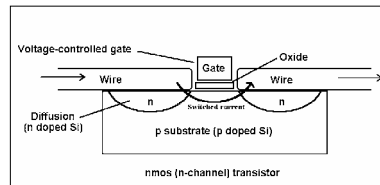
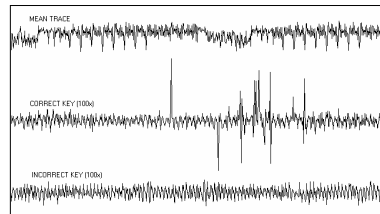
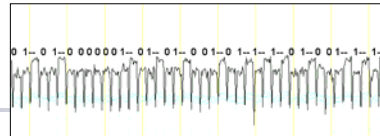


Glitchers sold for \$59-\$79 at <http://www.testsscards.com>



Simple and differential power analysis

- The attack:
 - Use a fast A/D converter to measure a target chip's power consumption
 - Analyze variations to find the keys
 - Powerful statistical techniques can extract keys from extremely noisy data
- Risk: High
 - Equipment is readily available and inexpensive; completely non-invasive
 - Fast and easy to repeat (typ. SPA <1 sec)
 - Evolving field: ~50% of papers at CHES conference deal with SPA/DPA attacks
- Countermeasures are very challenging
 - Analog issues, complex cryptography...
 - Difficulty increases dramatically w/ complexity
 - Large chips with multiple power/clock inputs often infeasible to protect fully
 - Testing is also very difficult

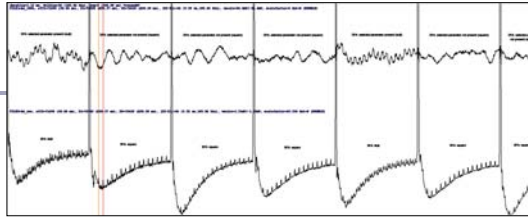


Patent notice: DPA Countermeasures are covered US patents including 6278783, 6298442, 6304658, 6327661, 6381699, 6510518, 6539092, 6654884, and foreign counterparts.



DEMA/SEMA attacks

- The attack:
 - Measure electromagnetic emanations to "see" inside an operating chips
- Risk: Moderate
 - Developing attack requires significant skill, but repetition is easy.
 - Equipment is moderately expensive
- Countermeasures are very challenging
 - Algorithmic countermeasures generally the same as for DPA
 - Chip-level shielding is very expensive
 - Normal chip packaging is not shielded
 - Packaging with custom metal cages is very expensive
 - Signals can still leak out (e.g., around pins, etc.)
 - Engineering skill set for countermeasure development is very specialized
 - Most research on RF characteristics of digital chips is focused on preventing interference (not security)



E&M measurements of a contactless smart card performing RSA



JTAG/SCAN attacks

- The attack:
 - Chips and boards have self-test capabilities
 - Needed for debug + post-manufacture test
 - Attackers use these test capabilities to extract keys and other data
 - Finding keys in scan chain is easy, e.g. a simple search
- Risk: Extreme if not fully addressed
 - Attack is in widespread use by criminals; requires only software + connector
- Countermeasures are very challenging
 - Remove test capabilities from all security-critical regions
 - Requires major changes to design flow, but gives the best security
 - Manufacturing test issues: Poor testing can lead to shipping chips with defects
 - Requires training: Most chip designers & fabs don't know how to do this safely
 - Use fuses or anti-fuses to disable test capabilities
 - Expensive - Requires manufacturing process with (anti-)fuse support
 - Security against invasive attacks is a matter of debate



\$39 interface cable for extracting keys from pay TV set top boxes via JTAG interface



Software bug exploitation

- The attack:
 - Almost all modern security chips are controlled by internal CPUs + firmware
 - Attackers can exploit bugs in the code to gain complete control of the device
- The risk: **Extreme**
 - Software bugs are the single most common cause of failure in data security systems
 - Attacks can be delivered via any data input (physical media, network/data connections...)
 - Note: Attacks usually target non-security code (more complex & not written by security experts)
- Countermeasures are very challenging
 - Sandboxing (SPDC, JavaScript...)
 - Limit what the software can do
 - Limit complexity
 - Testing (manual + automated) is essential, but isn't a full solution
 - Time consuming, expensive, misses many problems...
 - Recovery/renewal mechanisms to fix problems (limit damage)



TechNet Home > Security

Microsoft Security Bulletin MS04-028

Buffer Overrun in JPEG Processing (GDI+) Could Allow Code Execution (833987)

Summary

Who should read this document: Customers who use any of the affected operating systems, affected software programs, or affected components.

Impact of Vulnerability: Remote Code Execution

Maximum Severity Rating: Critical

Recommendations: Customers should apply the update immediately.

JPEG decoding buffer overflow bug announcement

Hacker bypasses Xbox firewall

DarkReading, CNN News.com

CNET News.com

April 01, 2006, 07:52 EST

Hacker bypasses Xbox firewall

An anonymous hacker has succeeded in running Linux on an unmodified Xbox, apparently satisfying a \$100,000 (plus 250,000) challenge issued by Linux founder, Richard Stallman.

A hacker using the name HabibiXbox revealed the exploit on Saturday in a message posted on the Xbox Linux Wiki site. Organizers of the Xbox-Linux Project confirmed the method worked.

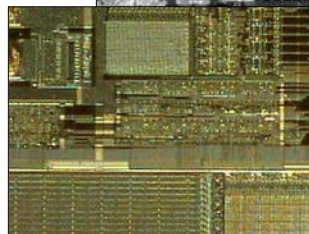
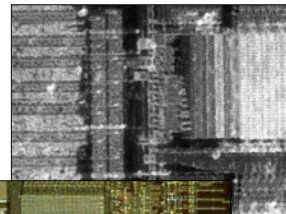
The trick involves the "save/load game" function in the James Bond game "007: Agent Under Fire", which normally allows players to save a file recording their progress in the game to the Xbox's hard drive and later reload it. HabibiXbox found that by using one of several "load" strings (codes recognized by the Xbox, the "load game" screen can allow to load or load other software, including compact versions of the Linux operating system.

The technique apparently exploits a "buffer overflow" flaw in the UDF game, a technique similar to that used by online vandals to damage servers. "Basically, there is a bug in the save handling, which has been found in several games," Habibi wrote in the posting.

Hackers have been working since shortly after the Xbox was released to modify the same. News article discussing an Xbox buffer overflow flaw

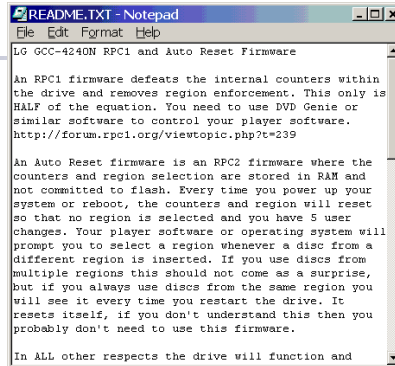
Hardware bug exploitation

- The attack:
 - Modern chips are designed using software languages (mostly VHDL & Verilog)
 - Attackers can exploit bugs in the hardware
- The risk: **Moderate (but increasing rapidly)**
 - Moderate to high; risk increasing as chips get more complex
 - Examples: Intel DIV bug, RAM cache bugs, etc.
- Countermeasures are very challenging
 - Mostly similar to software bugs:
 - Testing, limit complexity, sandbox high-risk components



Firmware modification

- The attack:
 - Software has to be stored somewhere. Attackers will try to modify the code where it is stored or as it is loaded.
- The risk: Extreme
 - Attacks are widespread today
- Countermeasures are very challenging
 - Use digital signatures or other cryptographic measures to try to secure the software
 - Must protect the verification itself from tampering
 - Challenge: updates & rollback attacks
 - Customized hardware can sandbox firmware or isolate cryptographic components
 - Major architectural change
 - Put firmware in ROM where it can't change
 - Cannot fix bugs



```
LG GCC-4240N RPC1 and Auto Reset Firmware

An RPC1 firmware defeats the internal counters within the drive and removes region enforcement. This only is HALF of the equation. You need to use DVD Genie or similar software to control your player software.
http://forum.rpc1.org/viewtopic.php?t=239

An Auto Reset firmware is an RPC2 firmware where the counters and region selection are stored in RAM and not committed to flash. Every time you power up your system or reboot, the counters and region will reset so that no region is selected and you have 5 user changes. Your player software or operating system will prompt you to select a region whenever a disc from a different region is inserted. If you use discs from multiple regions this should not come as a surprise, but if you always use discs from the same region you will see it every time you restart the drive. It resets itself, if you don't understand this then you probably don't need to use this firmware.

In ALL other respects the drive will function and
```

Readme.txt included with a firmware hack that bypasses region coding on a DVD drive



Quick recap

- Recap of the 8 attacks we reviewed:
 - Glitching (fault induction)
 - Timing attack cryptanalysis
 - Simple & differential power analysis
 - DEMA/SEMA attacks
 - JTAG/SCAN attacks
 - Software bug exploitation
 - Hardware bug exploitation
 - Firmware modification

These attacks are all non-invasive, cheap/easy to repeat,
... and require great skill to prevent

(The notion that hardware is inherently secure is a myth.)



Is it hopeless?

- Can we really address all of these?
 - Not hopeless – an experienced cryptographic hardware engineering team has good odds (~80%?) of success if...
 - ... they have determination, time, money, and skill
 - ... they successfully limit complexity
 - ... their work is carefully checked by experienced internal/external cryptographic validation teams
 - ... there are other systems that give attackers a better return for their effort
 - Example: Quality smart card vendors
- More implementations = more targets = much more risk
 - Having even a 10% failure rate isn't sufficient if there are 25 implementations available to attackers...



Conclusion: Take a realistic view

- Robustness rules can help
 - Increases implementation quality = more time between attacks
- Having a rule isn't enough:
 - Few vendors employ staff cryptographers experienced at preventing these attacks
 - But hundreds of students know how to mount the attacks
- Need to have plans for dealing with the full range of security failures and class attacks
 - Base strategy on practical, proven models
 - Robustness rules and defensive approaches must be complemented by effective renewability mechanisms



End