Lest We Remember: Cold Boot Attacks on Encryption Keys

J. Alex Halderman∗, Seth D. Schoen†, Nadia Heninger∗, William Clarkson∗, William Paul‡,
Joseph A. Calandrino∗, Ariel J. Feldman∗, Jacob Appelbaum, and Edward W. Felten∗

∗ Princeton University † Electronic Frontier Foundation ‡ Wind River Systems

17th USENIX Security Symposium 2008

Jan Kleine

Seminar: Computer Architecture

Background & Problem

Novelty

Key Ideas & Findings

Mechanisms & Implementation

Results

Proposed Solutions

Conclusion

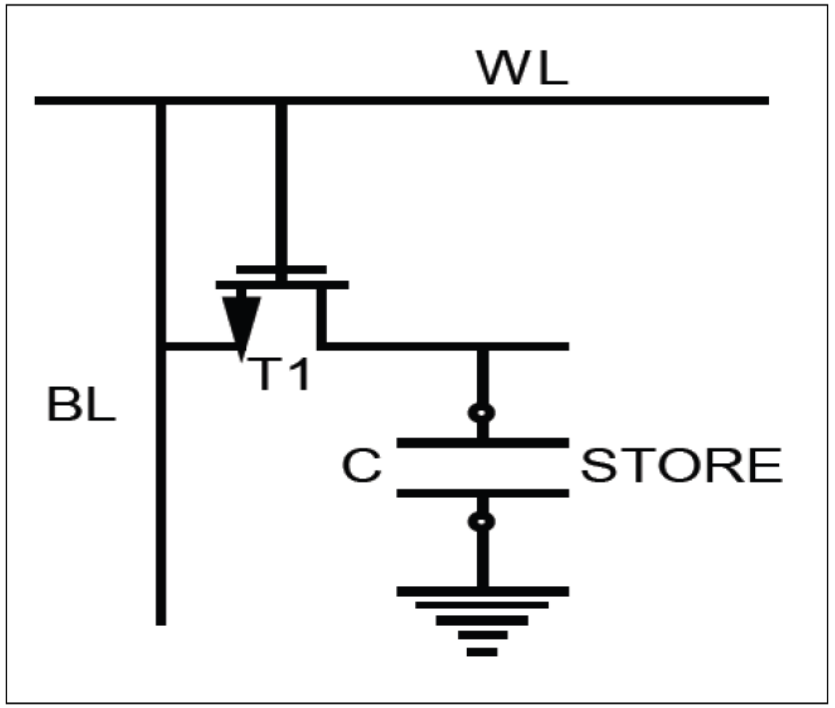
Strengths & Weaknesses

Follow-up Work

Discussion

A little refresh on DRAM

- A DRAM cell stores data in a capacitor
- Capacitor is charged to store data
- Capacitor loses charge over time
  - Ground state might be Vcc or Ground
  - Refresh (typically every 64ms)

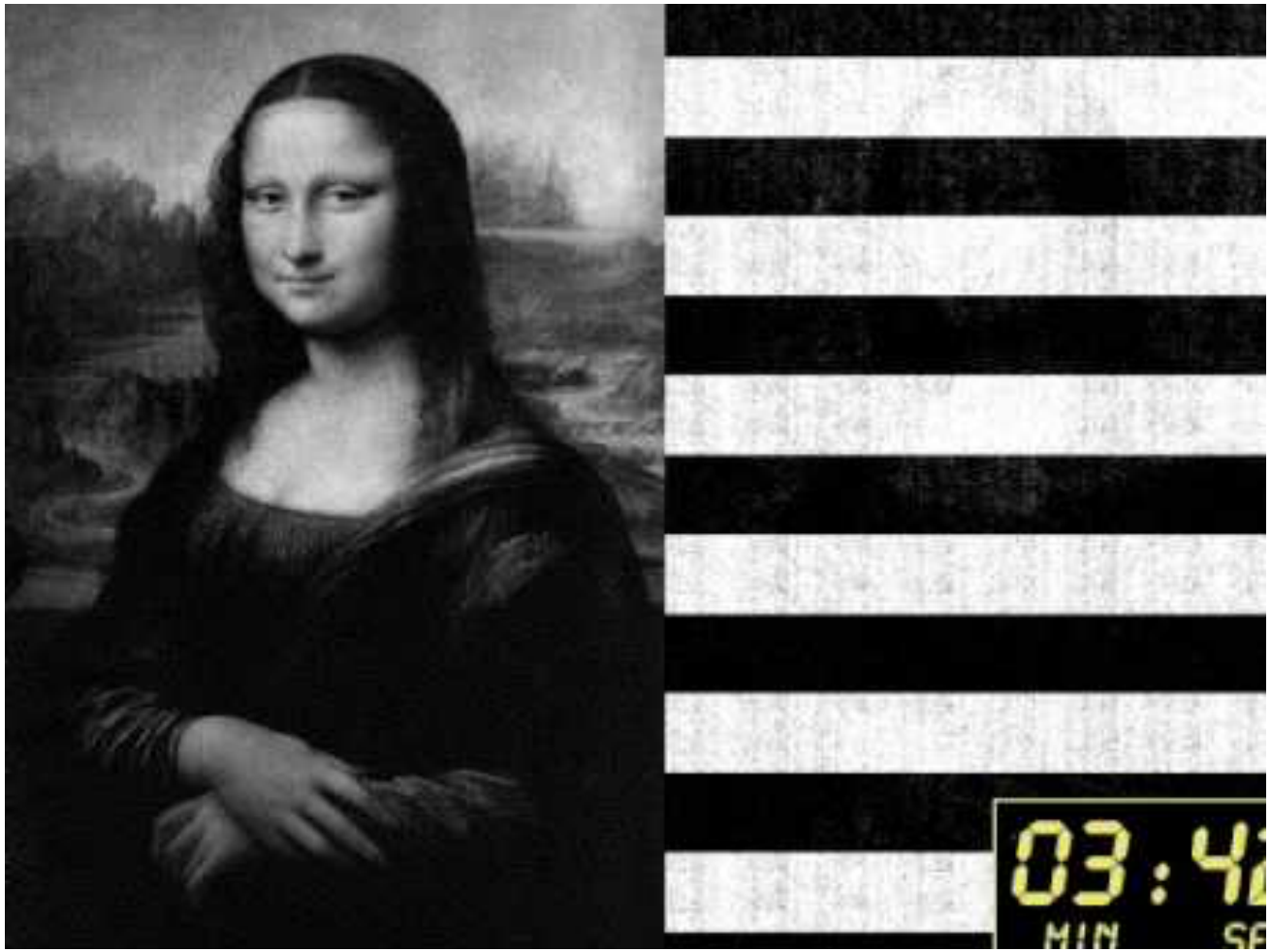


Problem

- DRAM retains data longer than refresh time
- Even after power loss, data is not lost right away
  - Data survives several seconds
- RAM is *the place* for sensitive data
  - Private secrets like encryption keys
- Potential for attacks on encryption keys and other sensitive data

Attacker can image memory and extract data after power loss

Problem: Example of data retention



Background & Problem

Novelty

Key Ideas & Findings

Mechanisms & Implementation

Results

Proposed Solutions

Conclusion

Strengths & Weaknesses

Follow-up Work

Discussion

Novelty

- First paper to demonstrate this attack vector of data retention
  - Demonstration on off-the-shelf hardware
- Novel algorithm for reconstructing corrupted encryption keys
- New method for finding (potentially corrupted) keys in memory

Background & Problem

Novelty

Key Ideas & Findings

Mechanisms & Implementation

Results

Proposed Solutions

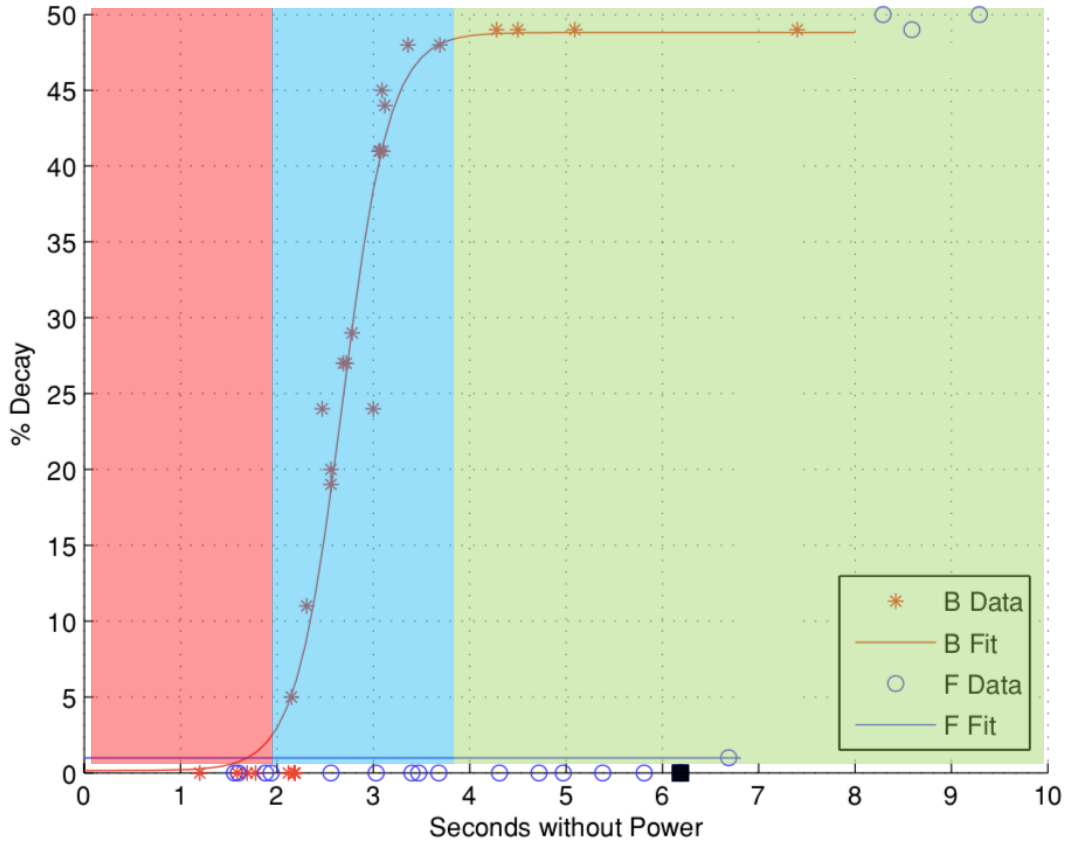
Conclusion

Strengths & Weaknesses

Follow-up Work

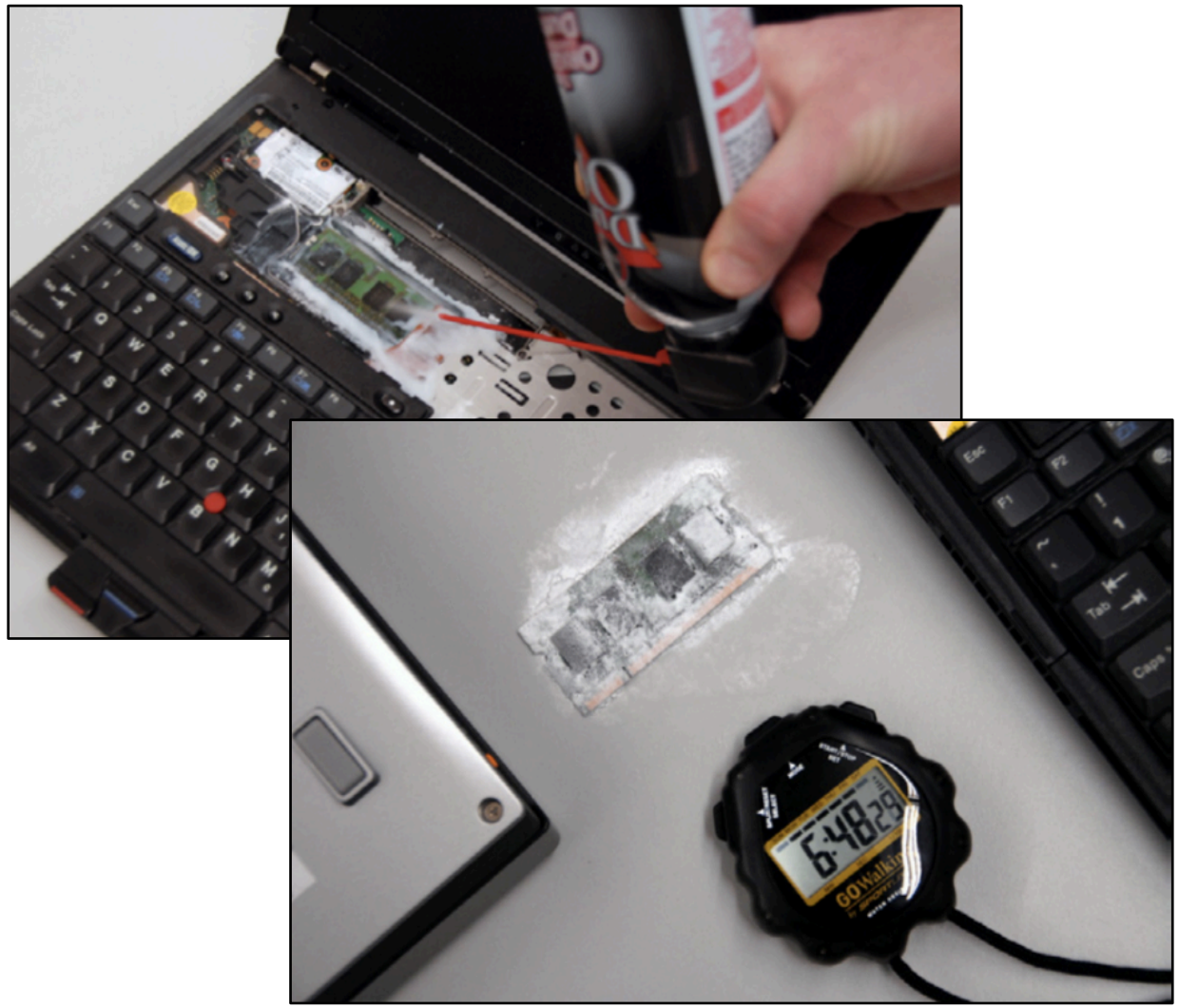
Discussion

DRAM Remanence



- Retention time is much longer than 64ms
- Most cells last in order of seconds
- Tested 6 modules from popular chip makers
  - Filled memory with pseudo random data
  - Measured loss after various times
- Period of little data loss followed by more rapid data loss
- Total data loss after 2.5 - 35 second

DRAM Remanence

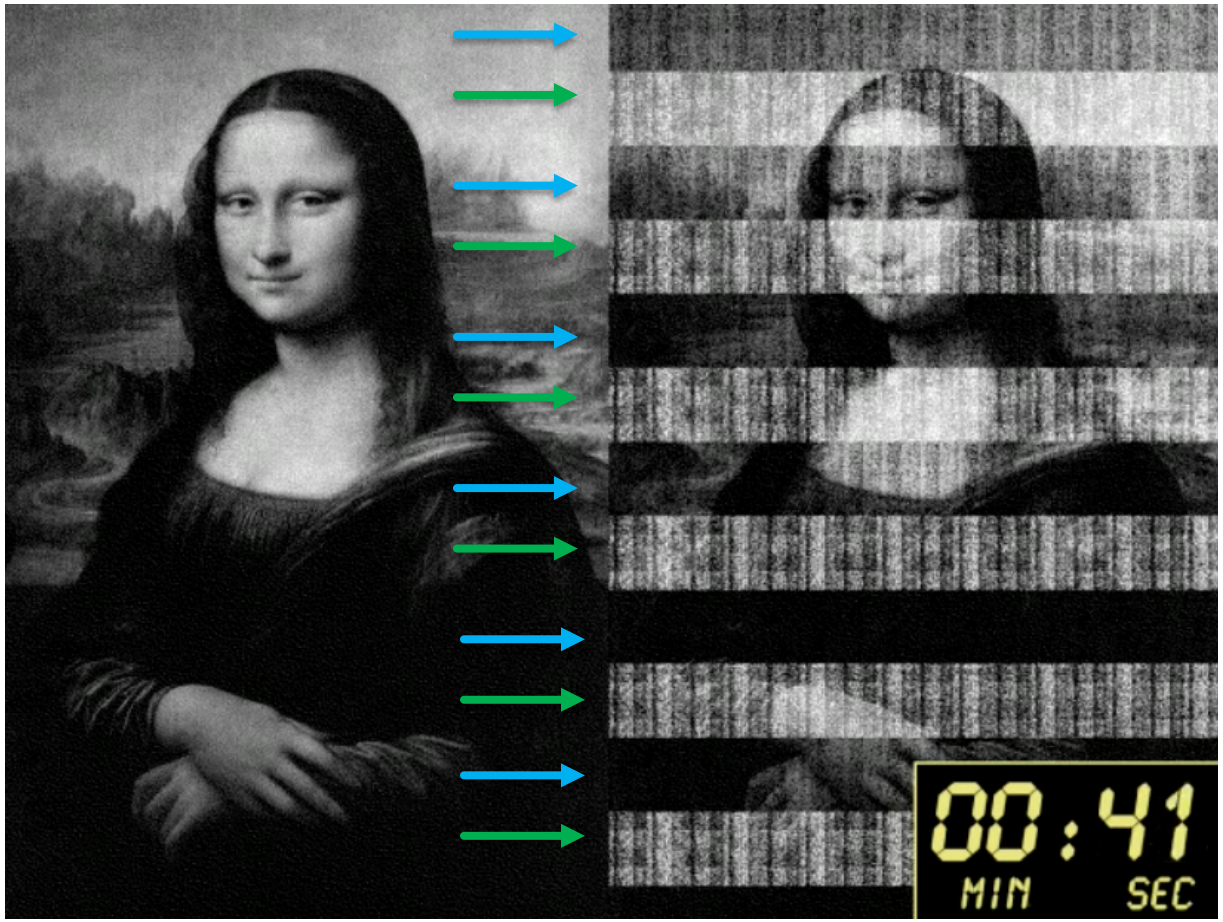


- Cooling down module drastically increases retention time
- At –50°C (Inverted air can)
  - All samples had ≥ 99.9% of data intact after 60 seconds
  - Typically ≤ 1% loss after 10 min
- At –196°C (Liquid Nitrogen)
  - 0.17% data loss after 60 min
- 1978 experiment showed **one week** without data loss[1]

1) W., AND MAY, H. Eigenschaften von MOS-Ein-Transistorspeicherzellen bei tiefen Temperaturen. Archiv für Elek*tronik und Übertragungstechnik 33* (June 1979), 229–235.

DRAM Remanence

- DRAM tends to decay in highly nonuniform pattern
- Relative order of decaying cells stays the same
  - regardless of temperature
- This makes the decay pattern predictable
  - Attacker can analyze decay pattern to reconstruct decayed data
- Striping pattern is explained by different ground states



Warm- and Cold-Boot Attack

Warm Boot

- Reboot machine from OS
- In BIOS select to boot from external drive
- Readout memory

Cold Boot

- Cut power to machine
- Restore power, start machine
- In BIOS select to boot from external drive
- Readout memory

Warm- and Cold-Boot Attack

Warm Boot	Cold Boot

Warm- and Cold-Boot Attack

Warm Boot	Cold Boot
Memory keeps refreshing, no data loss	

Warm- and Cold-Boot Attack

Warm Boot	Cold Boot
Memory keeps refreshing, no data loss	Memory briefly looses power, potential data corruption

Warm- and Cold-Boot Attack

Warm Boot	Cold Boot
Memory keeps refreshing, no data loss	Memory briefly looses power, potential data corruption
OS or application might overwrite data	

Warm- and Cold-Boot Attack

Warm Boot	Cold Boot
Memory keeps refreshing, no data loss	Memory briefly looses power, potential data corruption
OS or application might overwrite data	OS can’t overwrite data

Warm- and Cold-Boot Attack

Warm Boot	Cold Boot
Memory keeps refreshing, no data loss	Memory briefly looses power, potential data corruption
OS or application might overwrite data	OS can’t overwrite data
BIOS might erase memory	BIOS might erase memory

Warm- and Cold-Boot Attack

Warm Boot	Cold Boot
Memory keeps refreshing, no data loss	Memory briefly looses power, potential data corruption
OS or application might overwrite data	OS can’t overwrite data
BIOS might erase memory	BIOS might erase memory
BIOS and booted image will overwrite small portion of memory	BIOS and booted image will overwrite small portion of memory

Transferring DRAM module

- To avoid BIOS and kernel to overwrite memory transfer memory into other machine
- To aid transfer one would cool memory
- Attacker machine would create memory image

Background & Problem

Novelty

Key Ideas & Findings

Mechanisms & Implementation

Results

Proposed Solutions

Conclusion

Strengths & Weaknesses

Follow-up Work

Discussion

Basic Steps

I. Reconstructing corrupt keys
II. Identifying Keys in image
III. Imaging Residual Memory

I – Key Reconstruction

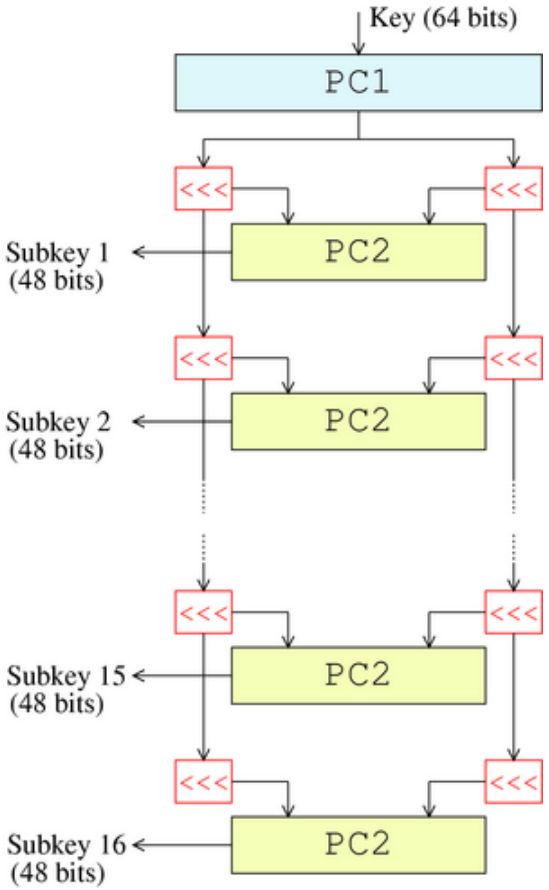
- Naïve way
  - Take recovered (corrupt) key and try keys with short hamming distance
  - Very easy to implement, but depending on error rate, not feasible
  - 12x 0→1 flip in a 256 bit key ⇒ 2^{56} different combinations ☹
- Target additional data as Error correcting code
  - Block cipher precompute sets of subkeys used for encryption (key schedules)
  - RSA uses extended private keys

Additional data makes reconstruction feasible

I – Reconstructing DES Keys

- DES key has 56 bits
- DES key schedule consists of 16 subkeys
  - Each a permutation of 48 bits
- Each bit is repeated about 14 times
  - Can be treated as repetition code
- LRW tweak keys can be reconstructed very similar
  - Used by TrueCrypt4

50% bit error can still be recovered with 98% chance

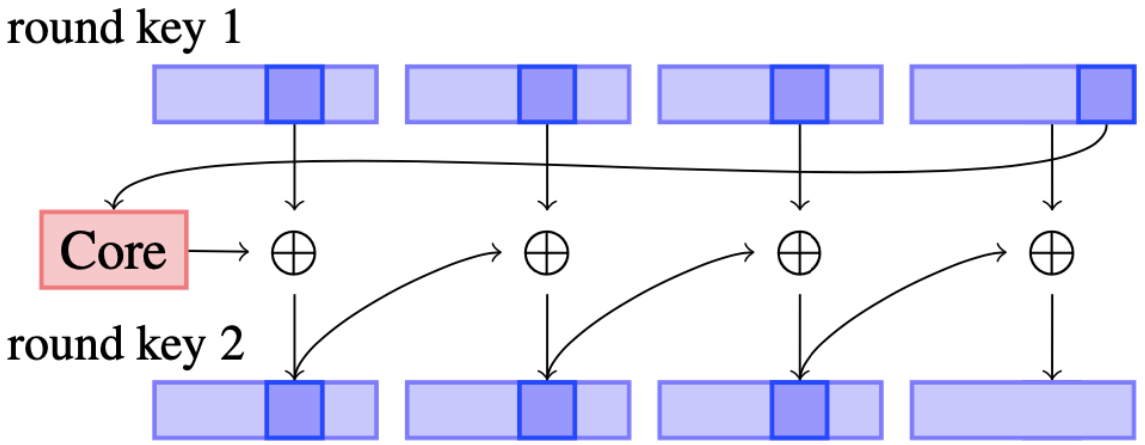


I – Reconstructing AES Keys

- Key schedule consists of 11 round keys
  - First one is key itself
- Reconstruction is more complicated then AES

1. Guess candidate keys from recovered table
2. Calculate key schedule
3. Check if the calculated schedule could have decayed to the recovered schedule

- XEX and XTS keys can be reconstructed this way as well (used by many programs)



15% bit error < 1 second
30% bit error ≈ 30 seconds

I – Reconstructing RSA Keys

- Public key consists of modulus N and public exponent e
- Private key consists of private exponent d and optional values
  - Prime factors p and q of N, $d\ mod\ (p-1)$, $d\ mod (q-1)$, $q^{-1} mod\ p$
- Public key components plus any one private component is enough
  - Chinese remainder theorem
- N can be factored in polynomial time given one of the following
  - $n/4$ least significant bits of p, d, or $d\ mod (p-1)$
- Alternatively ca… uess of $i-1$ lower bits of p a…

1024 bit key
4% error reconstructed in 4.5 seconds
6% error reconstructed in 2.5 minutes

II – Identifying Keys in Memory

- Approach is similar to Reconstructing
- Target additional data
  - Might be key schedules
  - Or predefined standards for key storage
- Older methods used statistical tests on memory
  - Check region of memory for statistical properties of encryption keys
  - Quick but many false positives

II – Identifying AES keys

- **`keyfind`** Algorithm
  - Iterate through each byte of memory and treat following 176-240 bytes as AES key schedule
  1. For each word in key schedule calculate Hamming distance to valid schedule
  2. If total number of bit violations is sufficiently small, output key
- Assumes key schedule in contiguous regions of memory
- Assumes byte order from AES specification
- This method is expected to work for many other ciphers (like DES)

II – RSA keys

- Two techniques

1. Search for know contents
   - Attacker is likely to know public modulus N
   - In case of Webserver it is obtained by querying the server
2. Find memory that matches DER encoding
   - DER is the standard for storing and interchanging private keys
   - Looking for features of DER encoding was very successful
   - No false positives

III – Imaging Residual Memory

- Custom software to boot from, which create memory image

1. Bootable USB drive that saves memory image
2. PXE network boot program that streams system memory via UDP
3. Installed memory imaging tools on an Apple iPod without impacting its functionality (how cool is that?)

Background & Problem

Novelty

Key Ideas & Findings

Mechanisms & Implementation

Results

Proposed Solutions

Conclusion

Strengths & Weaknesses

Follow-up Work

Discussion

Results

- They Demonstrated their attack on 5 different disk encryption programs
  - BitLocker
  - FileVault
  - TrueCrypt
  - dm-crypt
  - Loop-AES
- Fully automated tool for BitLocker
- Proof of concept for the others

Results – Bit(Un)Locker

- BitUnlocker, a fully automated tool to to unlock BitLocker disk encryption
  1. Cut power
  2. Boot from external disk with BitUnlocker installed
  3. BitUnlocker automatically creates memory image and runs **`keyfind`**
  4. Tries all candidate keys
  5. Mounts encrypted volume

25 minutes on ”modern” laptop with 2GB memory

Results – FileVault, TrueCrypt, dm-crypt, Loop-AES

- **`keyfind`** was able to automatically identify all AES keys for all programs
  - Keys did not contain errors
- FileVault uses secondary key which is also easily found in memory
- Loop-AES uses 65 different AES keys, all were found
  - Mapping the keys to specific disk blocks is easily done

Results – Other Interesting Findings

- In basic mode BitLocker loads key automatically into memory
  - Vulnerable even if computer shut down
- MacOS X (10.4 and 10.5) keep multiple copies of user password in memory
- Loop-AES stores an additional inverted copy of the key
  - Prevents memory burn-in
  - Makes key reconstruction easier

Background & Problem

Novelty

Key Ideas & Findings

Mechanisms & Implementation

Results

Proposed Solutions

Conclusion

Strengths & Weaknesses

Follow-up Work

Discussion

Countermeasures and their Limitations

- Scrubbing memory
- Avoid precomputation
- Key expansion
- Limiting boot options (BIOS password)
- Safe suspend

Countermeasures and their Limitations

- Physical defenses
  - Sensors that detect temperature changes and case openings
- Architectural changes
  - Deliberately reduce memory retention times
- Encryption in disk controller
  - Offload encryption to disk, OS doesn't hold key in memory
- Trusted Computing hardware
  - Hardware that monitors the boot process for suspicious behavior
  - Alternatively there can be a chip that hold encryption keys

Background & Problem

Novelty

Key Ideas & Findings

Mechanisms & Implementation

Results

Proposed Solutions

Conclusion

Strengths & Weaknesses

Follow-up Work

Discussion

Conclusion

- DRAM retention time is quite long and can be exploited
- The authors wanted to show the feasibility of this exploit
- To achieve this they found ways to…
  - image memory contents, without loosing keys
  - Find (potentially) corrupted keys in memory image
  - Reconstruct corrupt keys in reasonable time
- Build fully automated tool to break BitLocker
  - Showed that this attack can be applied to many other applications

Background & Problem

Novelty

Key Ideas & Findings

Mechanisms & Implementation

Results

Proposed Solutions

Conclusion

Strengths & Weaknesses

Follow-up Work

Discussion

Strengths and Weaknesses

Strengths

- Brought attention to the problem
  - Many follow-up papers and research
- Showed that it is hard to come up with a solution
- Shows attack in real world scenarios
  - Not just simulation
- Gives important data on real world DRAM modules
- Well written
  - Explains everything understandable

Weaknesses

- This is a physical attack, only applicable when you have physical access.
- Proposed solutions don’t really work

Background & Problem

Novelty

Key Ideas & Findings

Mechanisms & Implementation

Results

Proposed Solutions

Conclusion

Strengths & Weaknesses

Follow-up Work

Discussion

Follow-up work

Background & Problem

Novelty

Key Ideas & Findings

Mechanisms & Implementation

Results

Proposed Solutions

Conclusion

Strengths & Weaknesses

Follow-up Work, new Ideas

Discussion

Discussion

- What mitigations are used today/will be used in the future?
- What other mitigations can you think about
- Is the problem still relevant and will the problem stay relevant?
- What other uses do you see for this attack?