

# 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

USENIX Security Symposium, 2008

Presented by: Andra-Maria Ilies  
Seminar in Computer Architecture

# Executive summary

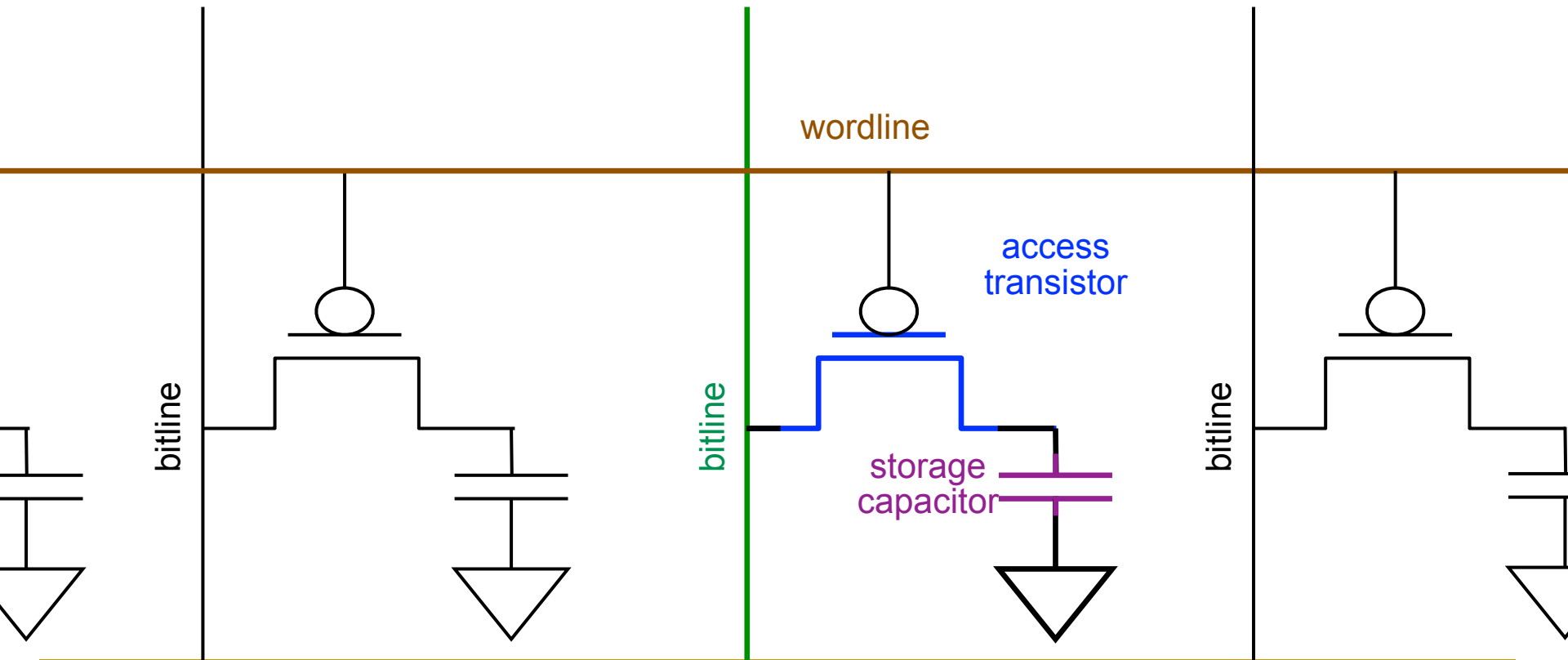
---

- **Problem:** DRAMs lose their data gradually after the power is cut
- **Goal:** Present a new type of attack which exploits remanence effect
- **Method:**
  - Acquire usable full-system memory image
  - Extract cryptographic key
  - Gain access to secret data
- **Evaluation:** succeeded on most popular disk encryption systems

# Background, Problem & Goal

# DRAM

- A DRAM cell consists of a **capacitor** and an **access transistor**.
- It stores data in terms of **change** in the capacitor.



# DRAM refresh

---

- DRAM capacitor charge **leaks** over time
- Each DRAM row is **refreshed** periodically to restore charge
  - Period usually is 64 ms
- **Retention time**: maximum time a cell can go without being refreshed while maintaining its stored data
- **Decay**: bit flips caused by charge leak
  - Cell leak = cell decays to **ground state**
- ~~When powered off DRAM loses its data completely~~

# Retention time and temperature

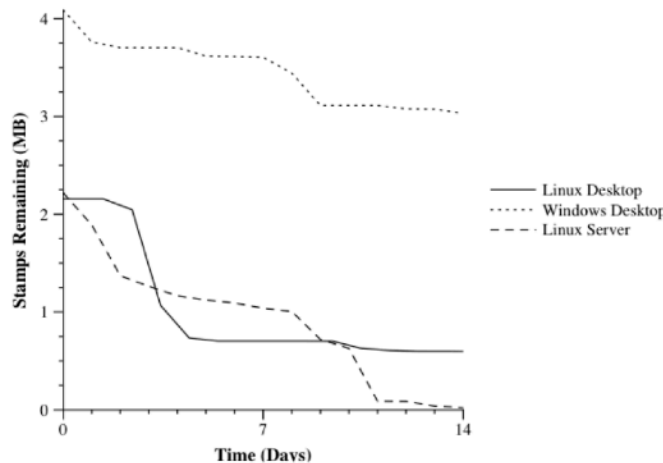
---

- Contents survive at some extent even at room temperature
- LINK, W., AND MAY, H. Eigenschaften von MOS - Ein Transistorspeicherzellen bei tiefen Temperaturen. Archiv für Elektronik und Übertragungstechnik 33 (June 1979), 229–235
- DRAM showed **no data loss** for **a full week** without refresh when cooled with liquid nitrogen
- Retention time can be increased by **cooling**

# Retention time and booting

---

- Chow, Jim & Pfaff, Ben & Garfinkel, Tal & Rosenblum, Mendel. (2005). Shredding your garbage: Reducing data lifetime through secure deallocation. USENIX 2005
- Experiment on **data lifetime**
- On **soft reboot** some data remain in memory
- On **hard reboot** results varied
  - Once laptop kept some data for **30s** after hard reboot



# Problem & Goal

---

## Problem

- DRAM data is still **available after powered off**
- Retention time can be made longer by cooling
- This gives enough time to an attacker to capture the memory

## Goal

- Exploit the **remanence property** of DRAM
- Mount attack on disk encryption systems
- Bypass disk encryption by obtaining encryption key

# Novelty

# Novelty

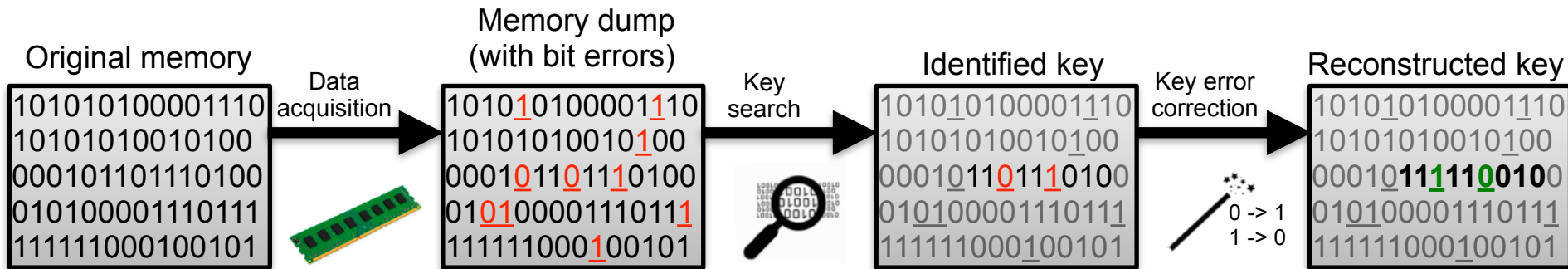
---

- Exposes a **new type of physical attack**
- First security study with focus on **security** implications of **DRAM remanence**
- New method to obtain memory image
- New algorithm for **reconstructing keys** in the presence of errors
- First to apply attacks on real **disk encryption** systems
- First to offer systematic discussion of **countermeasures**

# Key approach and Ideas

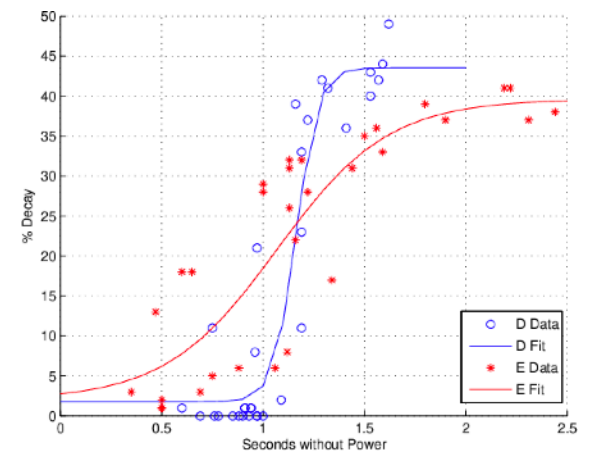
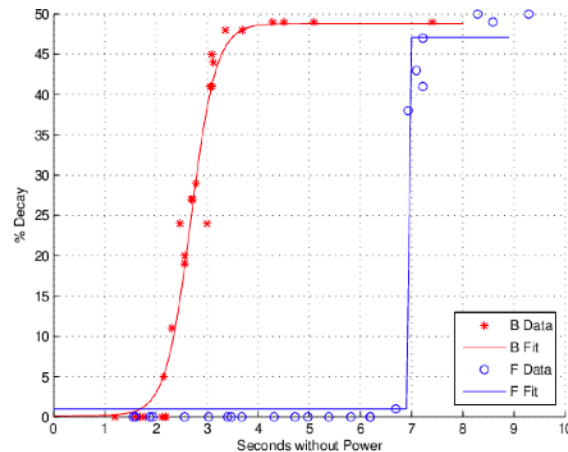
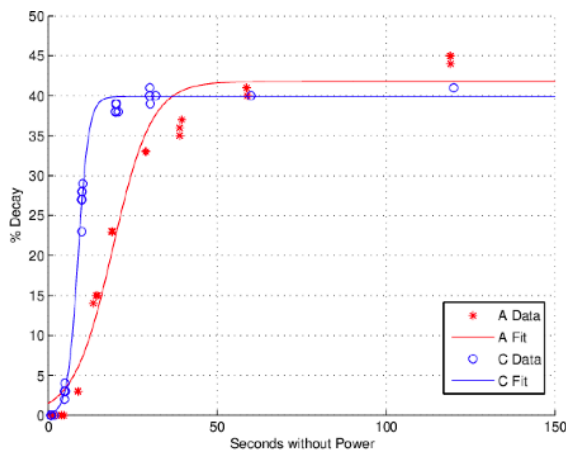
# Key approach

- Steps of cold-boot attack:
  - ➔ 1. Extract memory
  - ➔ 2. Locate key in memory
  - ➔ 3. Reconstruct decayed keys
  - ➔ 4. Decrypt hard drive



# Decay at operating temperature

- Method:
  - Full memory with pseudorandom pattern
  - Read back these regions after various periods of time
- Without refresh
- **Observation**: decay curves are similar
  - Initial period of slow decay, intermediate period of rapid decay, final period of slow decay



# Decay at reduced temperature

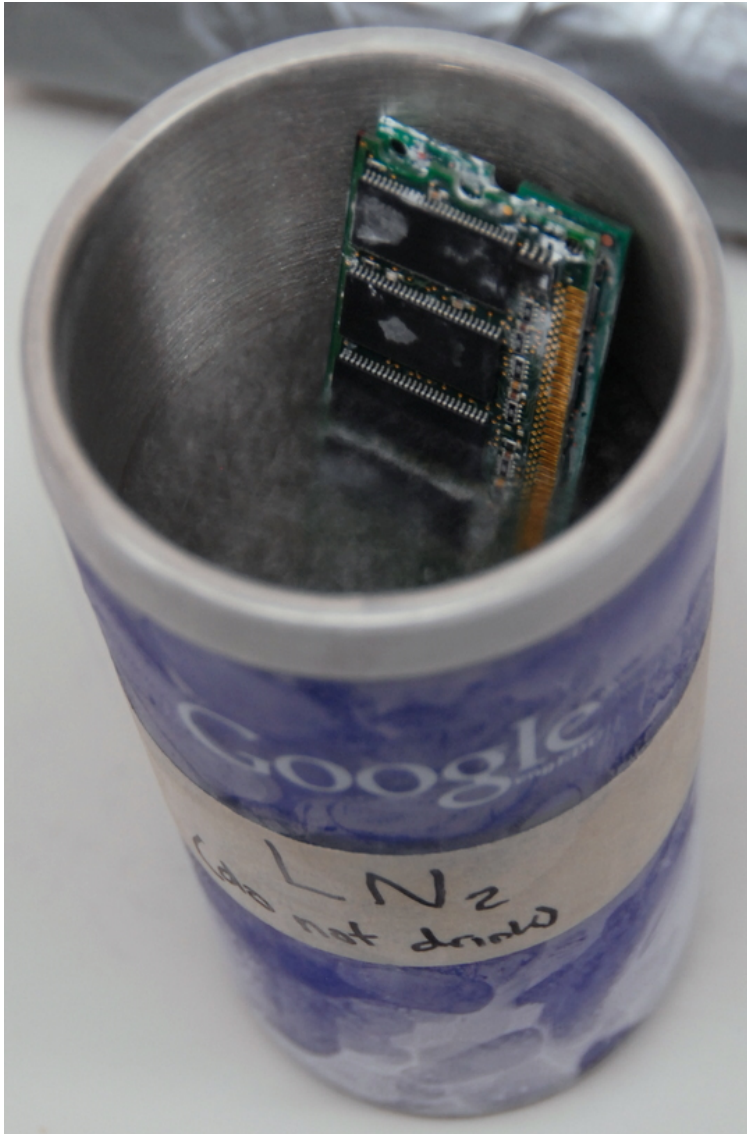
- Method:
  - Load pseudorandom test pattern
  - Cool down to **-50°C** using compressed air
  - Power off machine and maintain temperature
  - Restore power



	Seconds w/o power	Error % at operating temp.	Error % at -50 °C
A	60	41	(no errors)
	300	50	0.000095
B	360	50	(no errors)
	600	50	0.000036
C	120	41	0.00105
	360	42	0.00144
D	40	50	0.025
	80	50	0.18

# Decay at reduced temperature

---



- Use liquid nitrogen
- $-196^{\circ}\text{C}$
- **<0.17%** decay after **1 hour**

# Decay patterns and predictability

---

- DRAM tends to decay in **non-uniform patterns**
- **Patterns** and **order** are predictable
- Almost all bits tend to decay to **predictable ground**



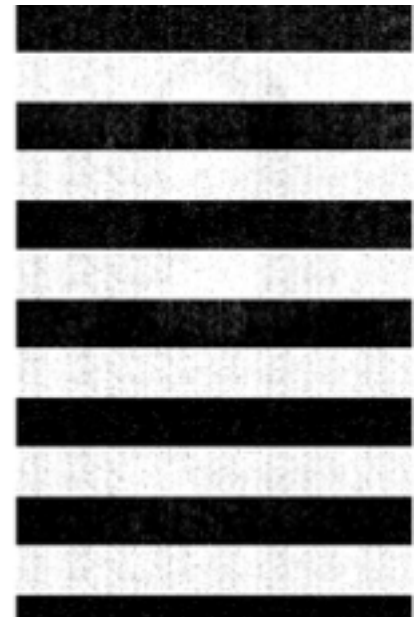
5 sec



30 sec



60 sec



5 min

# Mechanisms

# Imaging tools

---

- Used to produce dumps of memory to external medium
- Preboot Execution Environment(PXE) network boot
- USB drives
- Extensible Firmware Interface(EFI) boot
- iPod

# Imaging attacks

---

- **1. Simple reboot**

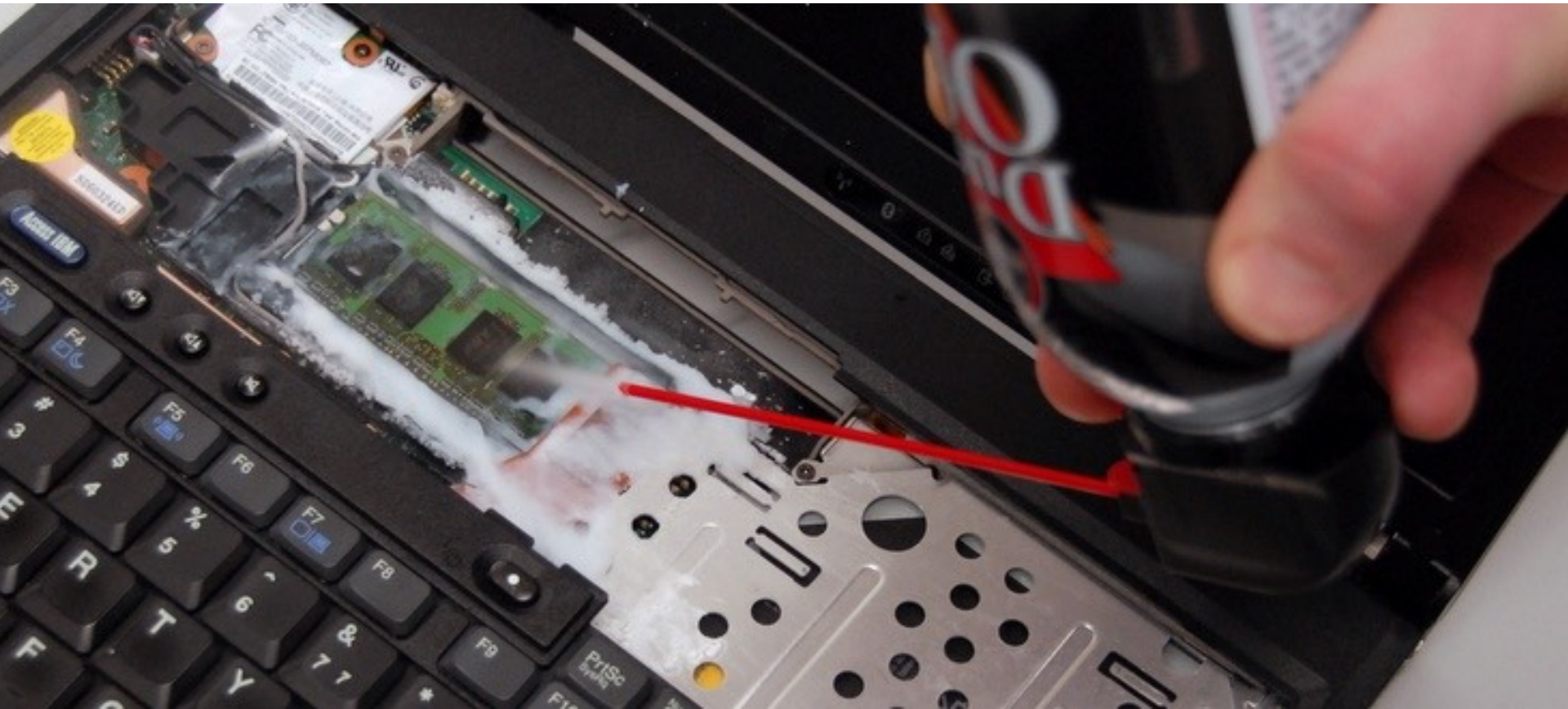
- Reboot machine and configure BIOS to boot from imaging tool

- **2. Transferring memory module**

- Physically remove DIMM
- Capture image using another computer
- Slow decay by cooling

# Slowing decay by cooling

---



**<0.2% decay after 1 min**





# Identifying keys in memory

---

- **Brute force**

- Large key space
- Presence of **bit errors** makes it intractable

- **Fully automatic** techniques to locate keys in memory **in presence of bit errors**

- Target **key schedule**
  - Key schedule uses multiple round keys derived from a single original key to modify intermediate result
- Search **blocks of memory** that satisfy combinatorial properties of a valid key schedule

# Key schedule

---

- Exploit the fact that most encryption programs speed up computation by **storing precomputed data** from encryption key
  - ➔ AES - key schedule with 1 sub-key for each round(12-14)
  - ➔ RSA - extended form of private key,  $p$ ,  $q$
- This data contains more **structure** than key by itself
- All the studied disk encryption systems precompute key schedules and keep them in memory for as long as the encrypted disk is mounted

# Identifying keys in memory: AES

---

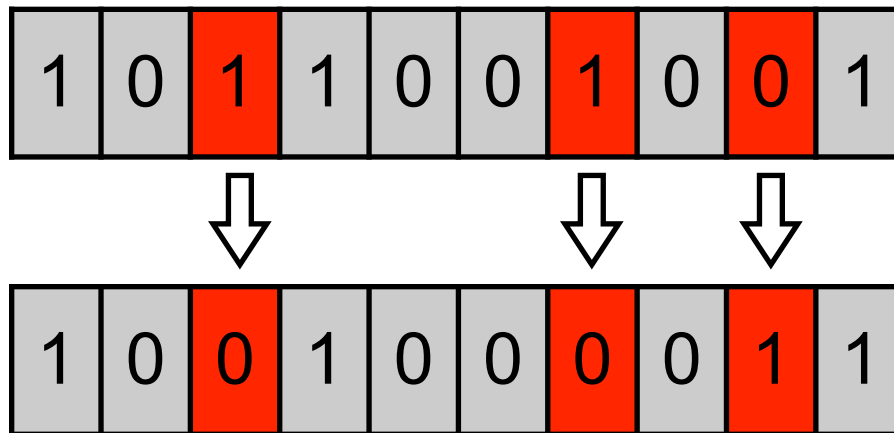
- **Input:** memory image
- **Output:** list of keys
- **Algorithm:**
  - ➔ **1.** Iterate through each byte of memory. Treat the following **block** of 176 or 240 bytes as a AES key schedule
  - ➔ **2.** For each words in the potential key schedule, calculate the **Hamming distance** from that word to the key schedule word that should have been generated by the surrounding words
  - ➔ **3.** If the total number of bits violating the constraints on a correct AES key schedule is **sufficiently small**, output the key.

# Key reconstruction 1: Brute force

---

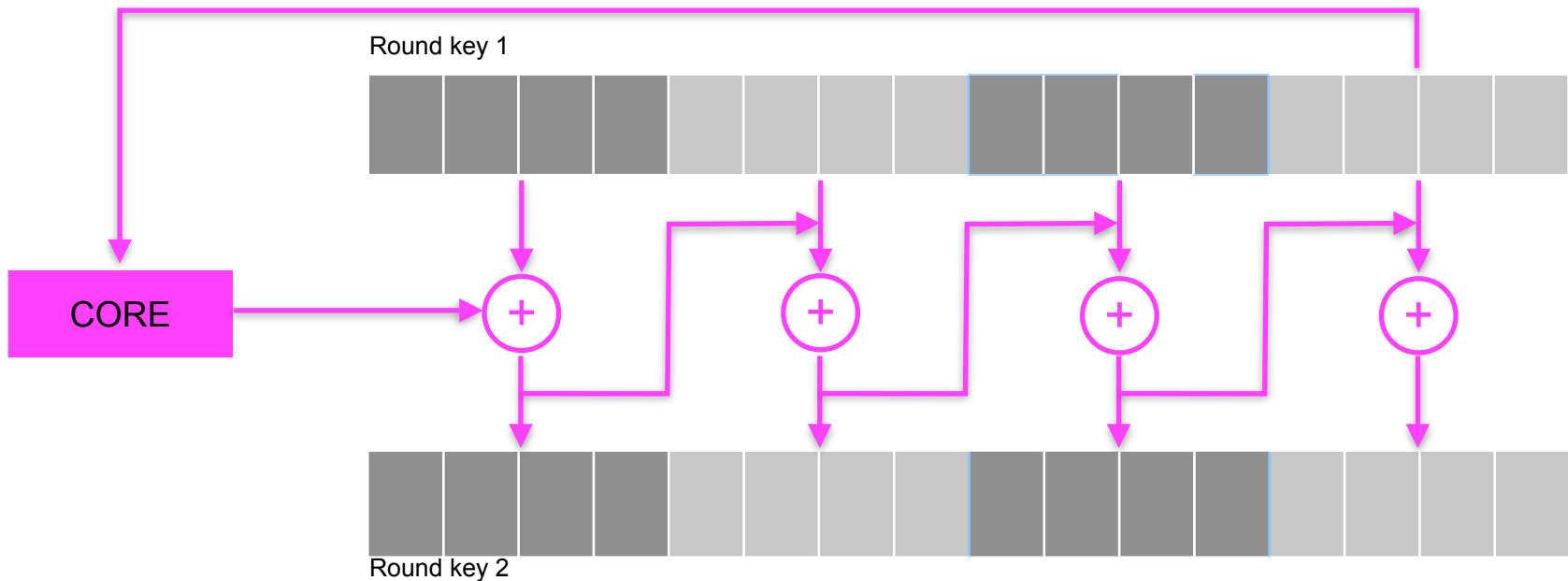
- Perform **error correction** on key
- Brute force key over keys with a low **Hamming distance** from the decayed key that was retrieved from memory
- (-) computational burden
  - 10% of 1s decayed => possible keys  $> 2^{56}$

**Hamming distance = 3**

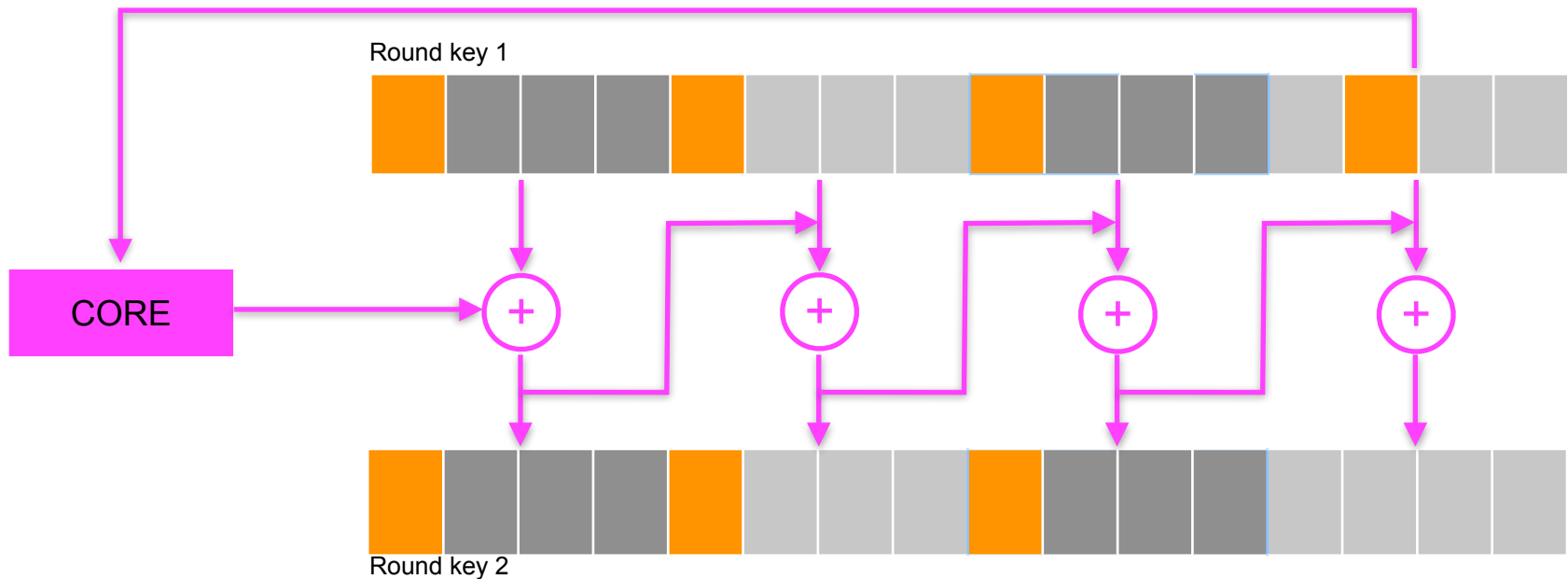


# Key reconstruction 2: AES key

- Exploit structure of AES key schedule:
  - ➔ Brute force segments
  - ➔ Combine to form key
- 128 bit key  $\rightarrow$  11 128-bit round keys

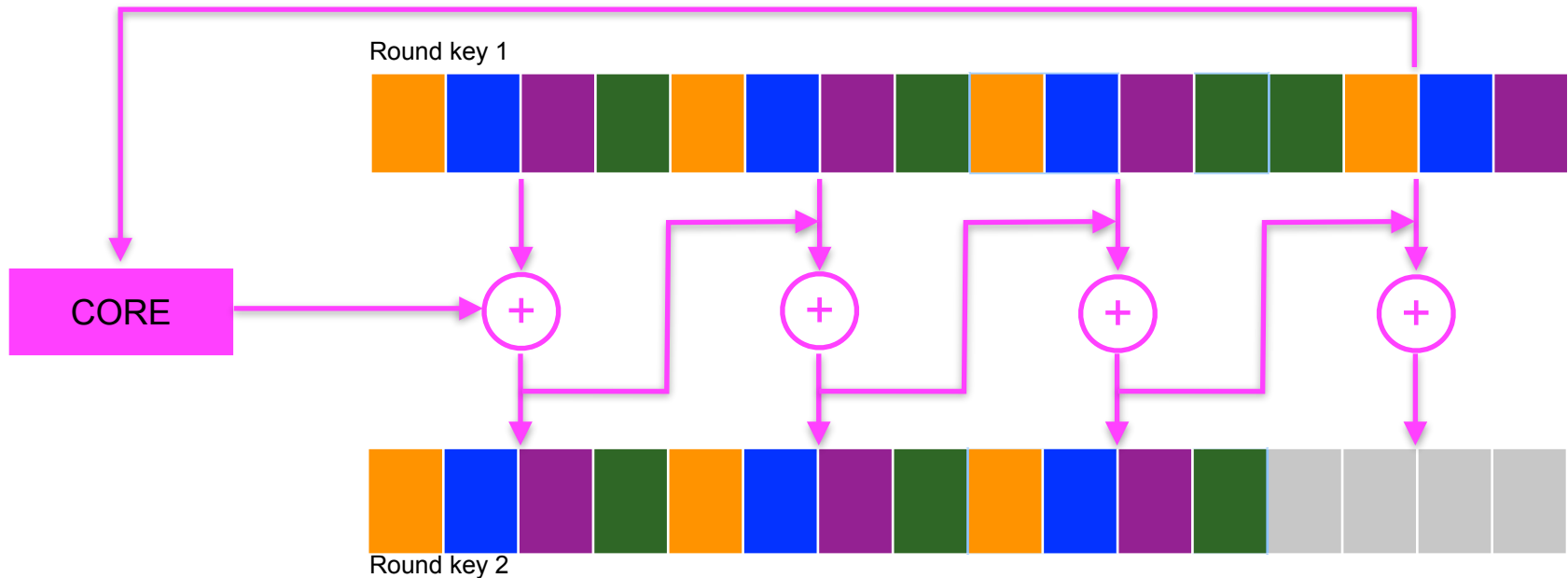


# Key reconstruction 2: AES key



- **1.** Slice: 4 bytes in Round  $n$  determine 3 bytes in Round  $n+1$
- **2.** Examine each  $2^{32}$  possibility in order of distance to recovered key
- **3.** Calculate the probabilities that the bytes decayed

# Key reconstruction 2: AES key



- **4.** Repeat for all 4 slices
- **5.** Combine in candidate keys (calc. probability of decay)
- **6.** Test candidates keys by expanding them into full key schedules – compare to recovered memory

# Key Results: Methodology and Evaluation

# Methodology

---

	Memory Type	Chip Maker	Memory Density	Make/Model	Year
A	SDRAM	Infineon	128Mb	Dell Dimension 4100	1999
B	DDR	Samsung	512Mb	Toshiba Portégé	2001
C	DDR	Micron	256Mb	Dell Inspiron 5100	2003
D	DDR2	Infineon	512Mb	IBM T43p	2006
E	DDR2	Elpida	512Mb	IBM x60	2007
F	DDR2	Samsung	512Mb	Lenovo 3000 N100	2007

Table 1: Test systems we used in our experiments

# Evaluation

---

- Performed the attack on most popular disk encryption systems
  - BitLocker
  - File Vault
  - TrueCrypt
  - Dm-crypt
  - Loop-AES

# FileVault

---

- 128-bit AES in CBC mode
- **2 keys:**
  - AES keys
  - Key to compute initialisation vector



- **Result:**
  - ✓ Attack recovered **only AES key**
  - ✓ Can decrypt 4080 bytes out of 4096 in a disk block
  - ✓ Can use previous methods to obtain initialisation vector key

# Loop-AES

---

- **On-the-fly** encryption
- 128-bit AES using option “multi-key-v3”
  - Each disk block is encrypted with one of 64 encryption keys
- **Result:**
  - ✓ The 64 keys found
  - ✓ Assignment between keys and blocks with trial decryptions
- Stores key schedule and also an inverted copy
  - Protection against memory burn-in
- For attacker this is useful additional redundancy

# Evaluation

---

Disk encryption is **valuable**,  
BUT  
**not** necessarily a **sufficient** defence

# Countermeasures

---

- **Scrubbing memory**
  - Overwrite keys when not in use
  - Clear memory at boot time
  - (-) still can physically move the memory to different computer with a more permissive BIOS
- **Limiting booting from network or removable media**
  - Require password
  - (-) swap out drive
  - (-) easy to reset NVRAM to re-enable booting from external device

# Countermeasures

---

- **Suspending a system safely**

- Power off machine when not in use
- Guard machine after powered off
- (-) inconvenient

- **Avoid precomputation**

- (-) hurts performance

- **Key expansion**

- Apply some transform to key as it is stored in memory
- Key is more resistant to reconstruction

# Countermeasure

---

- **Physical defences**

- Lock DRAM modules on machine
- System could respond to cold temperatures
- (-) additional cost

- **Architectural changes**

- Add key-store hardware that erases state on power-up, reset, shutdown
- (-) old machines still at risk

# Countermeasures

---

- **Encrypt in the disk controller**

- Use a write-only **key register** for encryption
- (-) key register is now vulnerable

- **Trusted computing**

- Boot history decides if it is safe to keep key in RAM
- (-) once key is in RAM, system is vulnerable

# Summary

# Summary

---

- DRAM holds values surprisingly long after powered off
- This enables security attacks
- Steps:
  - ➔ **1. Extract memory** - data decay slowed down by cooling
  - ➔ **2. Locate key in memory** - target key schedule's redundancy
  - ➔ **3. Reconstruct decayed keys** - target key schedule's redundancy
  - ➔ **4. Decrypt hard drive**
- Disk encryption systems which use various encryption techniques are vulnerable
- Many countermeasures, but each has its tradeoffs
- Disk encryption is not enough to protect against a physical attack

---

Questions?

# Strengths

# Strengths

---

- Thorough study of misconceptions about DRAM
- Opened research towards a new type of attacks
- Defeated the most commonly used disk encryption products
- Works for both symmetric and asymmetric encryption
- Fast
- Non-destructive
- Requires accessible equipment
- Open source tools and [demo](#)
- Well written, easy to read
  - Especially well analysed countermeasure section

# Weaknesses

# Weaknesses

---

- Paper assumes that all bits decay to the same ground state
- Analyse attack at “more normal” temperatures
- Attack on each system requires system specific tools
- Key identification assumes that key schedules are contained in continuous regions of memory

# Thoughts and Ideas

# Thoughts and ideas

---

- Does it work for other devices?
  - Paper focuses on laptops
- Attach a boot monitoring tool and wait for CPU to request sensitive data
- Introduce some randomness in key schedule storage

# Takeaways

# Takeaways

---

- What is a cold boot attack and how it can be performed
- Encryption is not as secure as it seems
- ~~Data fades instantaneously when DRAM has no power~~
- ~~Residual data is difficult to recover~~
- Temperature influences decay speed
- Leakage happens as a result of computation
  - But can happen also when no computation is done
- Example of tradeoff between security and performance

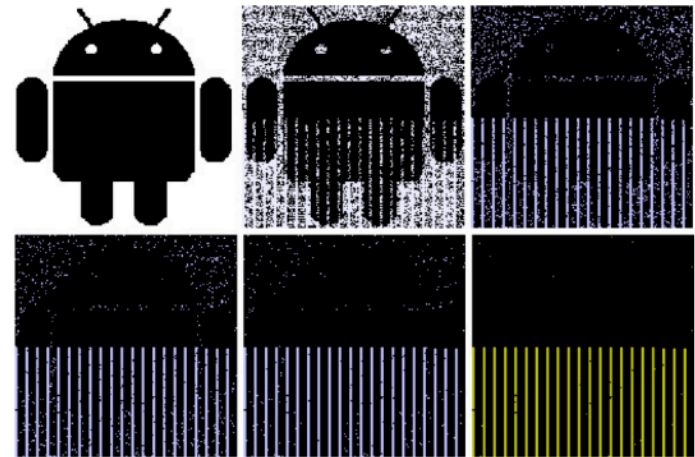
# Open Discussion

# Open discussion

- What other devices could be susceptible to this attack?
- Mobile phones
  - Müller, Tilo & Spreitzenbarth, Michael. (2013). FROST: forensic recovery of scrambled telephones

	$\varepsilon$	0.5 – 1s	1 – 2s	3 – 4s	5 – 6s
5 – 10 °C	0 (0%)	2 (0%)	1911 ( 5%)	8327 (25%)	24181 (73%)
10 – 15 °C	0 (0%)	976 (2%)	2792 ( 8%)	18083 (55%)	25041 (76%)
15 – 20 °C	0 (0%)	497 (1%)	4575 (13%)	20095 (61%)	25433 (77%)
20 – 25 °C	0 (0%)	421 (1%)	16461 (50%)	23983 (73%)	27845 (84%)
25 – 30 °C	1 (0%)	2204 (6%)	16177 (49%)	27454 (83%)	28661 (87%)

**Fig. 5:** Number of bit flipping errors per physical page (in total and percentage) dependent on the phone temperature and the time of battery removal.



**Fig. 8:** An Android bitmap after 0s, 0.5s, 1s, 2s, 4s, and 6s in DRAM without power. The cold boot attack has been deployed at room temperature.

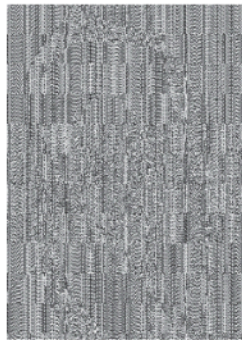
# Open discussion

---

- Is this still an issue nowadays?
- Mitigation: **Data scrambling**
  - XOR'ing it with a pseudorandom number before writing it to DRAM
  - Yitbarek, Salessawi & Aga, Misiker & Das, Reetuparna & Austin, Todd. (2017). Cold Boot Attacks are Still Hot: Security Analysis of Memory Scramblers in Modern Processors



(a) Original Image



(b) Scrambled DDR3 Data



(c) Scrambled DDR3 Data  
Read Back After Reboot



(d) Scrambled DDR4 Data



(e) Scrambled DDR4 Data  
Read Back After Reboot

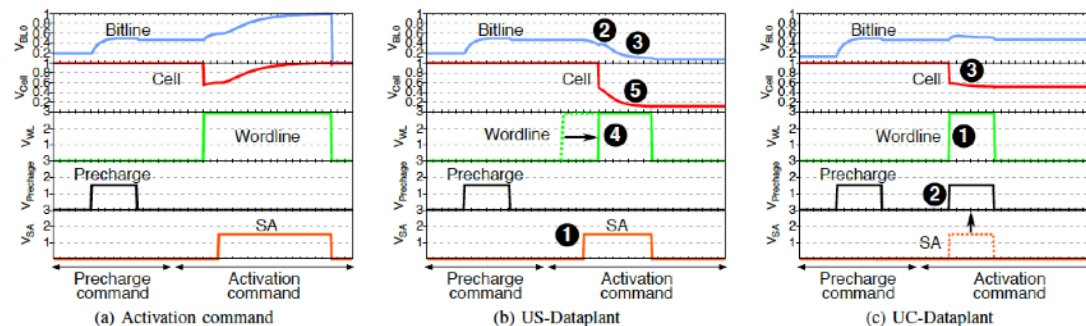
# Open discussion

---

- Is this still an issue nowadays?
- Mitigation: **Overwrite content of DRAM**
  - For performance reasons this is not done at every start
- Attack
  - **Clear firmware bit** for memory overwrite request
  - Settings stored on non-volatile memory
  - [Attack demo](#) and [blogpost](#)

# Open discussion

- Is this still an issue nowadays?
- Mitigation: **clear DRAM data at startup**
  - Orosa, Lois & Wang, Yaohua & Puddu, Ivan & Sadrosadati, Mohammad & Razavi, Kaveh & Gómez-Luna, Juan & Hassan, Hasan & Mansouri-Ghiasi, Nika & Tavakkol, Arash & Patel, Minesh & Kim, Jeremie & Seshadri, Vivek & Kang, Uksong & Ghose, Saugata & Azevedo, Rodolfo & Mutlu, Onur. (2019). Dataplant: Enhancing System Security with Low-Cost In-DRAM Value Generation Primitives
  - Mechanism completely implemented in DRAM by **changing the internal DRAM timing signals**
  - Depends on power-on detection circuit
  - Solutions to cold boot attack:
    - **Self destruction** - refresh the whole DRAM memory in self-refresh mode at power-on, using Dataplant primitives instead of activation commands
    - **Command based destruction** - memory controller forces DRAM to obey sequence of instructions that leads to data destruction at power-on



# Open discussion

---

- How could the encryption process be changed?
- Müller, Tilo & Freiling, Felix & Dewald, Andreas. (2011). TRESOR runs encryption securely outside RAM
  - Take advantage of Intel's new AES-NI instruction set
  - Exploits the x86 **debug registers** in a non-standard way, namely as cryptographic key storage.
- Exploit **variation in retention time** of DRAM cells
  - Store key in a part of memory with less retention time

# Open discussion

---

- Other data that could be obtained in a similar way?
- [Naveed, Muhammad & Ayday, Erman & Clayton, Ellen & Fellay, Jacques & Gunter, Carl & Hubaux, Jean-Pierre & Malin, Bradley & Wang, Xiaofeng. \(2014\). Privacy in the Genomic Era. ACM Computing Surveys](#)
- **Genomic data** is different than traditional healthcare data
  - Properties: health/behaviour, static, unique, mystique, value, kinship
- Privacy risks:
  - Re-identification threats
  - Phenotype inference: aggregate genomic data, correlation of genomic data, kin privacy breach
  - Other: anonymous paternity breach, legal and forensic
- Users generally not equipped with skills and equipment to protect the security and privacy of their genomic data
  - Solution: **store it on a cloud** in an encrypted fashion, such that attacker needs to circumvent cloud security
- Data sharing issue
  - Solution: **functional encryption** - computation directly on encrypted data

# Open discussion

---

- Could it be possible to protect information even if key is leaked?
- Leakage Resilient Cryptography
  - Regev, Oded. (2005). On Lattices, Learning with Errors, Random Linear Codes, and Cryptography. Journal of the ACM (JACM)
  - Moni Naor and Gil Segev. 2009. Public-Key Cryptosystems Resilient to Key Leakage. In Proceedings of the 29th Annual International Cryptology Conference on Advances in Cryptology (CRYPTO '09)

# Open discussion

---

- Are DRAM alternatives at risk?
- NVRAM?
  - Attack is trivial
- Hybrid?
  - What would be kept on DRAM and what on NVRAM?

# 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

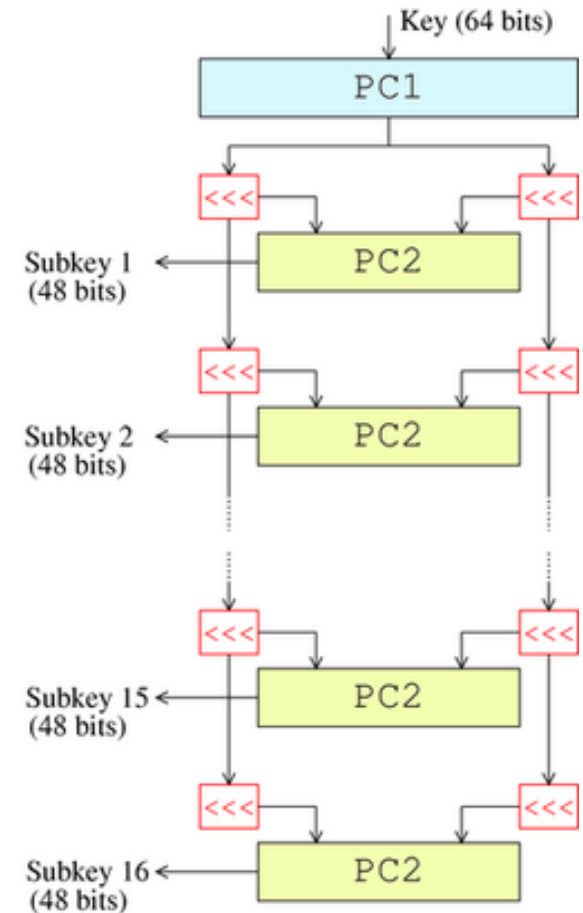
USENIX Security Symposium, 2008

Presented by: Andra-Maria Ilies  
Seminar in Computer Architecture

# Backup slides

# Key reconstruction: DES

- Exploit information from **key schedule**
- DES key schedule:
  - 16 subkeys
  - Each subkey is a permutation of 48-bit from the original 56-bit key
  - Every bit from the original key is repeating in 14/16 sub-keys



The key-schedule of DES

Treat DES key scheduler as a **repetition code**

# Key reconstruction 2: DES key

---

- Treat DES scheduling as a **repetition code**
  - The message is a single bit, and the corresponding codeword is a sequence of N copies of this bit
- Notation:
  - $\delta_0$  - probability of a 1 flipping to 0
  - $\delta_1$  - probability of a 0 flipping to 1
- If  $\delta_0 = \delta_1 < \frac{1}{2} \rightarrow$  optimal decoding of bit is 0 if more than  $n/2$  recovered bits are 0, else is 1  $\rightarrow$  **max occurrences**
- If  $\delta_0 \neq \delta_1 \rightarrow$  optimal decoding is 0 if more than  $N \cdot r$  of the recovered bits are 0, else is 1

$$r = \frac{\log(1 - \delta_0) - \log \delta_1}{\log(1 - \delta_0) + \log(1 - \delta_1) - \log \delta_1 - \log \delta_0}$$

# Results: key reconstruction

---

- DES
  - Even at 50% error, probability of key being correct **>98%**
  
- AES
  - Reconstruct key with 15% error in **fractions of a second**
  - Reconstruct half of keys with 30% error in **30 s**
  
- RSA
  - 1024-bit primes
    - Error 4% - **4.5 s**
    - Error 6% - **2.5 min**
  - 512-bit primes
    - Error 10% - **1 min**