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

Presented by Tom Yurek

Background: How does DRAM keep state?

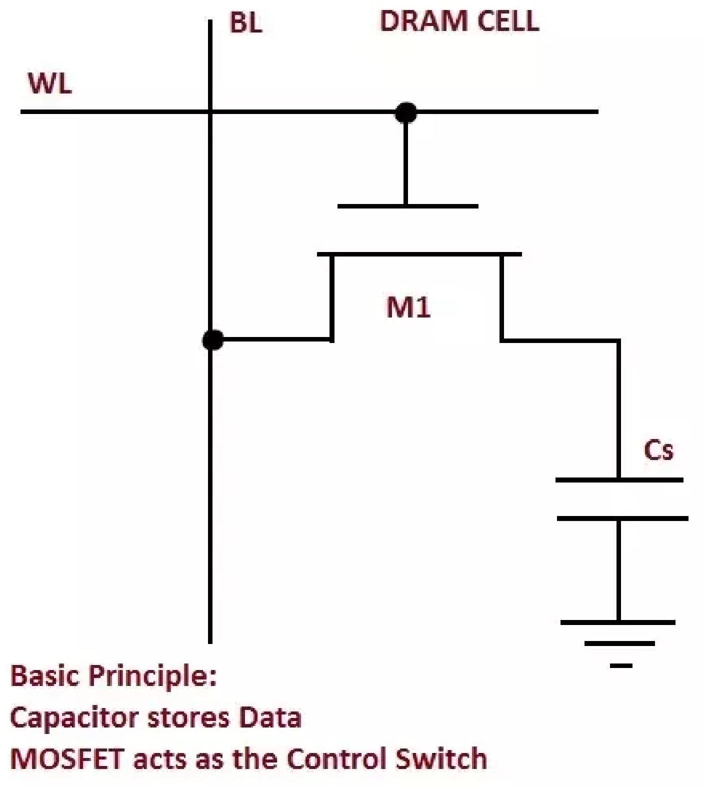


Diagram Credit: AllThingsVLSI

WL: Word Line, determines whether or not a cell should be written to

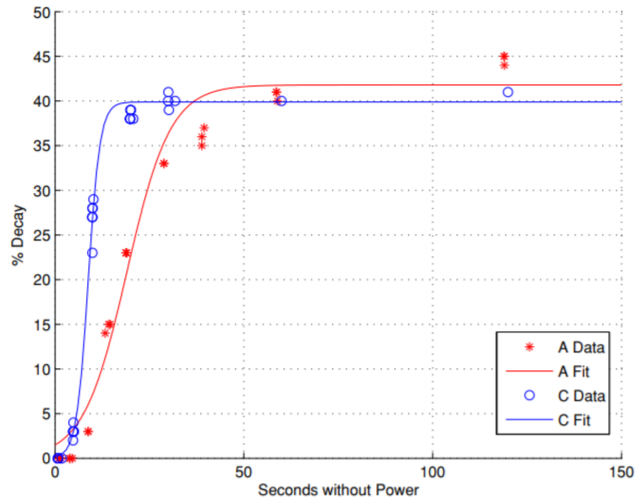
BL: Bit Line, determines which value to write

Capacitor Cs is charged for a 1, discharged for a 0

State is refreshed on the order of milliseconds (DDR2 spec had a max refresh period of 64ms)

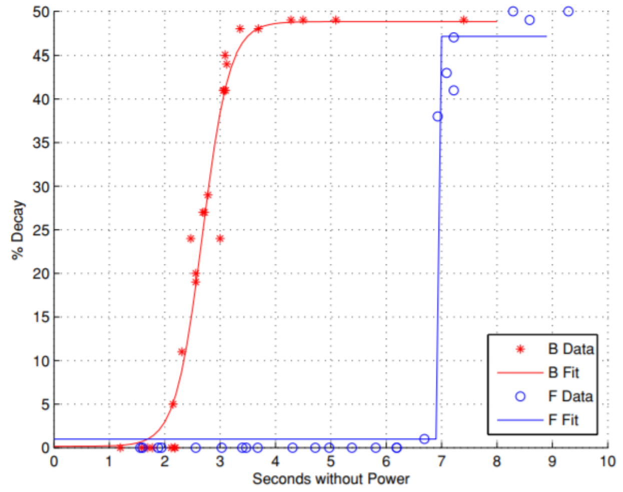
Cs may be attached to power or ground depending the cell’s address

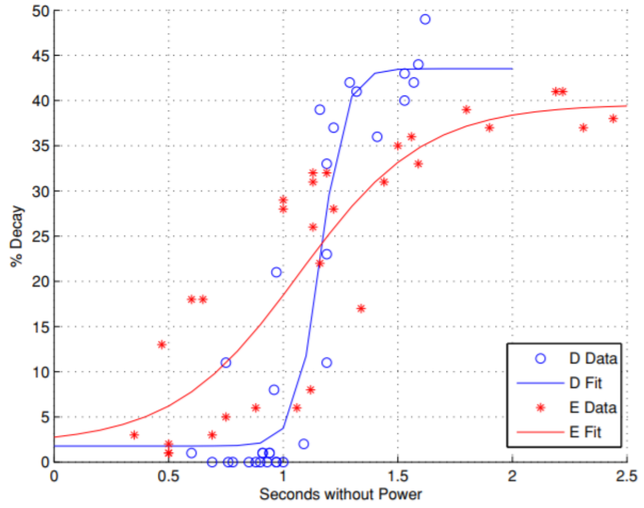
Fast discharge assumptions are invalid for most cells



DRAM decay rate tested for 6 machines at room temperature

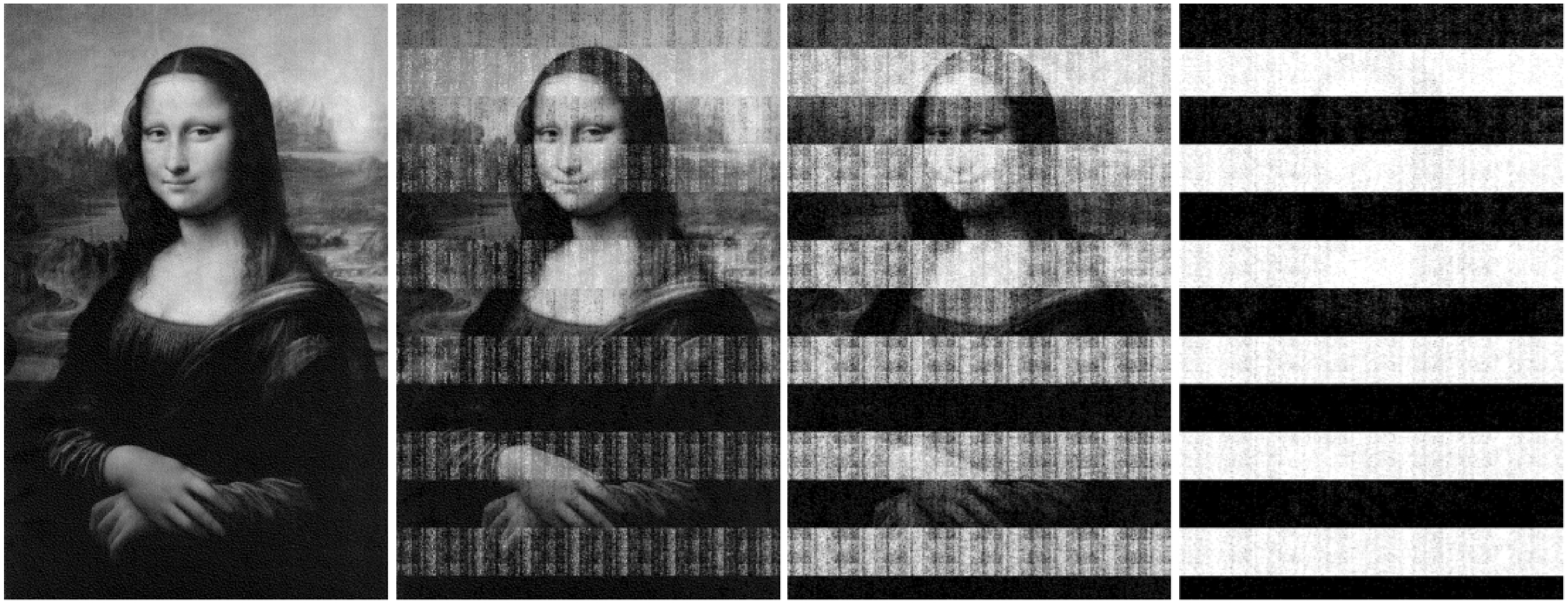
Decay took between 1.5 to 35 seconds with older and less dense memories taking longer.





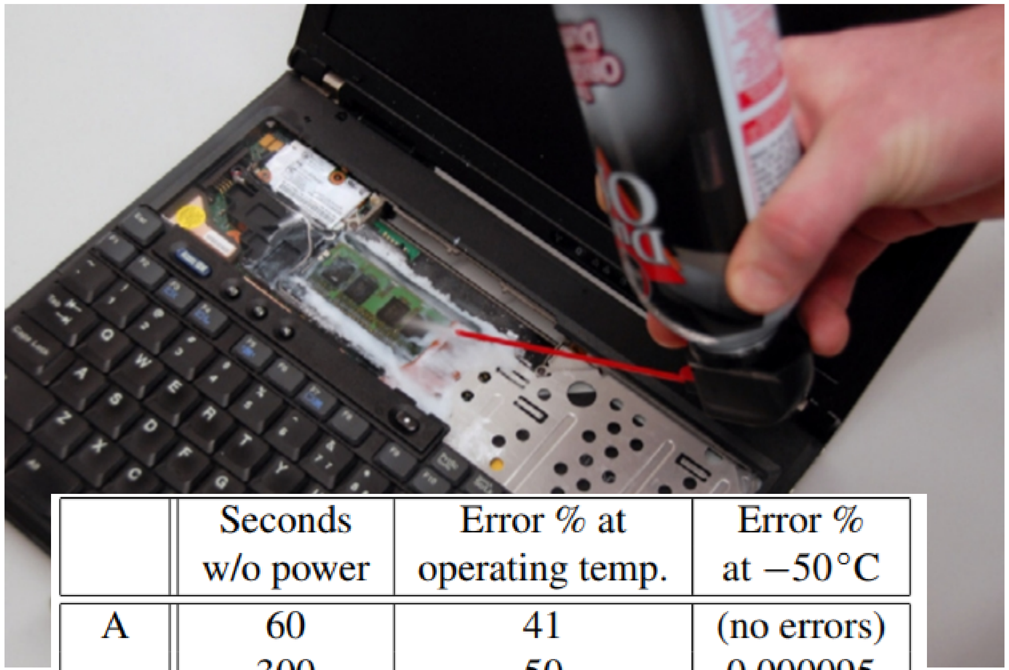
A slower decay rate is still desirable to reduce power consumption and improve performance

An Image Representation of Decay



5 seconds 30 seconds 60 seconds 5 minutes

What if you blast it with compressed air first?



	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

A much more guaranteed attack is to cool the memory with compressed air first, before shutdown.

A memory module cooled in liquid nitrogen for an hour experienced .17% decay.

Existing Hurdles for Preserving Memory

The simplest and least effective way to perform a cold boot attack is to restart the computer and boot into a custom kernel to analyze memory.

Cutting the power is better, but BIOS may still overwrite parts of RAM.

In this work, the same device is used to read memory. A tiny, single-purpose OS was created solely for memory dumping.

Okay, but is it enough to extract keys?

If even one bit of a 256 bit key is incorrect, it would take 256 guesses to brute force the correct key. In this case, an error rate of approximately 5% (about 13 bits) would require on the order of 10^21 guesses to resolve. And that's assuming you know exactly where the key is….

$$256 C\, 13 = \frac{256!}{10!*243!} = 2.3895*10^{21}$$

Solution: Use the Key Schedule

Many crypto implementations will store intermediate values of the key in memory, called a key schedule.

Every disk encryption system the authors tested stored precomputed key schedules in memory.

Using this information, a key can be feasibly recovered with 1->0 error rates between 5 and 50 percent.

Validation against known ciphertexts is not needed!

Key Extraction Attack Against DES

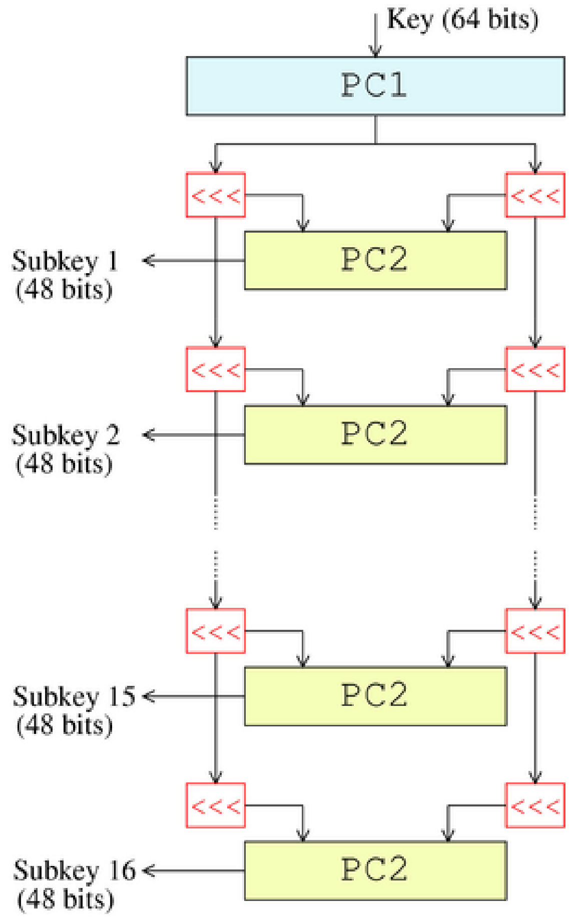


Image Credit: Wikipedia

In DES, 16 subkeys are produced, each with a permutation of 48 out of 56 bits of the key. So each bit of the key appears in 14 subkeys.

Combine occurrence rates and known flip probabilities to find most likely key

The paper's attack can be effective even when half of the 1's in memory have decayed to 0's

Key Extraction Attack Against AES

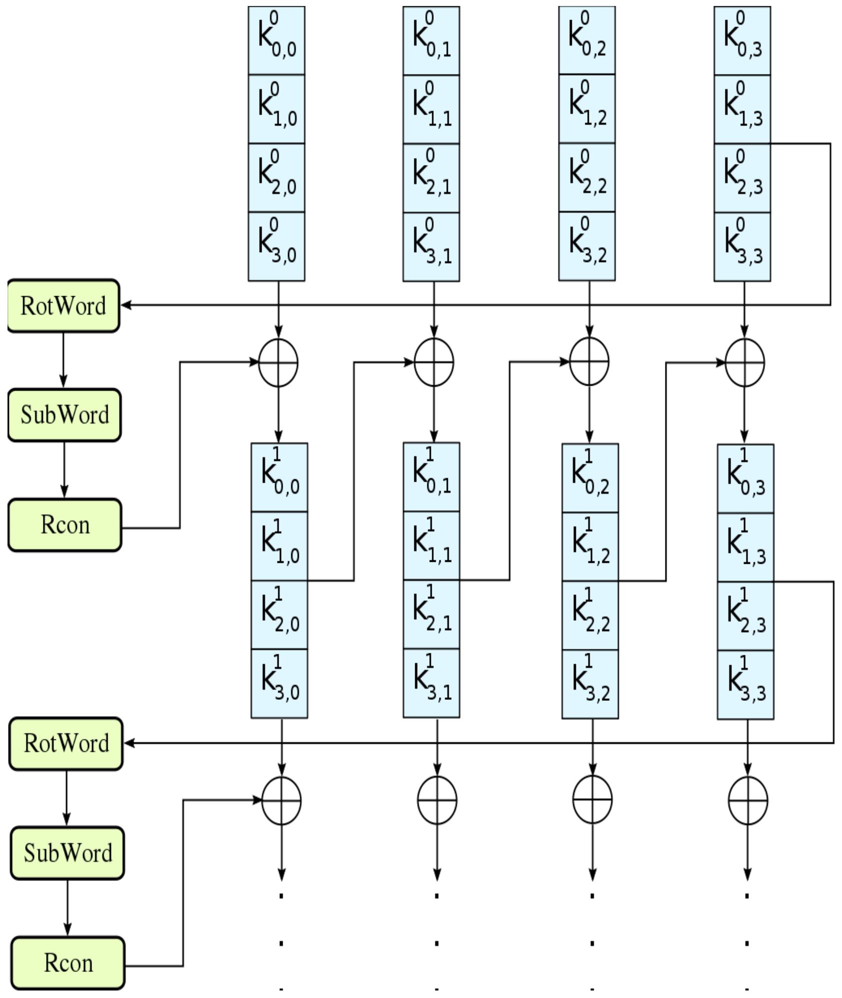


Image Credit: Wikipedia

The AES key schedule is more sophisticated, with 11 keys in the schedule, each derived from the last by a function of XORs, a rotation, S-Box substitutions, and exponentiation of 2 in a finite field.

Key is divided into parts and each likely part is checked against the key schedule

The authors also find key recovery attacks against tweakable encryption and RSA.

But how do you find the keys?

Brute forcing over 1GB of RAM to test each possible 128 bit key aligned to 4-byte machine words would require 2^28 tests. This becomes impractical once memory errors are introduced.

Like earlier, use the key schedule.

Search for parts of memory that satisfy or are close to satisfying key schedule properties.

Authors use this method to find keys from black box crypto implementations and claim that this outperforms bit entropy scans of memory.

How does that actually work?

AES:
For each word of memory, test whether the next 176 or 240 bytes of memory could be an AES key schedule.
Take the first 128 bits and determine if the next 128 bits represent a key within a certain hamming distance of expected.
Continue throughout the schedule.

RSA:
You could search memory for known information, such as a public modulus
Authors also found success by searching memory for data encoded in the way specified by the PKCS #1 spec.

Attacks Against Drive Encryption

BitLocker: Often included in Vista, uses two AES keys, a sector pad key and CBC encryption key encrypt the whole disk. Immediate rebooting is not even needed if using TPM, as it reloads the keys in RAM upon startup.

FileVault: OSX's disk encryption tool keeps the AES key in memory, IVs can be found afterwards. Multiple copies of the user's login password were also found in memory.

TrueCrypt, dm-crypt, and Loop-AES were also broken with extra security features often making the attack easier.

Countermeasures

- Scrub memory when you’re done with it
- Limit booting from other media
- Suspend the system safely
- Don’t precompute Key Schedules
- Transform the key to make it hard to reconstruct with errors present
- Physically protect DRAM
- Architectural Changes
- Use the disk controller for encryption
- Use (better) trusted computing