The Cold Boot Attack

Nadia Heninger

November 20, 2009

Joint work with: J. Alex Halderman, Seth D. Schoen, William Clarkson, William Paul, Joseph A. Calandrino, Ariel J. Feldman, Rick Astley, Jacob Appelbaum, and Edward W. Felten

The Persistence of Memory

For a good time, run

```
sync
```

then run something like the following Python program until you get bored:

```
a = ""
while 1: a += "ARGON"
```

Then yank the power on your computer, reboot, and

```
sudo strings /dev/mem | less
```

Do you find any `ARGON`?

(Extended instructions at `citp.princeton.edu/memory/exp/`)

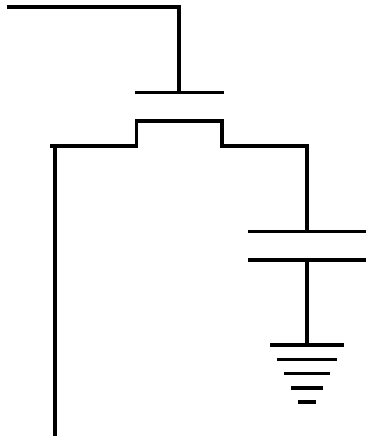
The Persistence of Memory: Why?

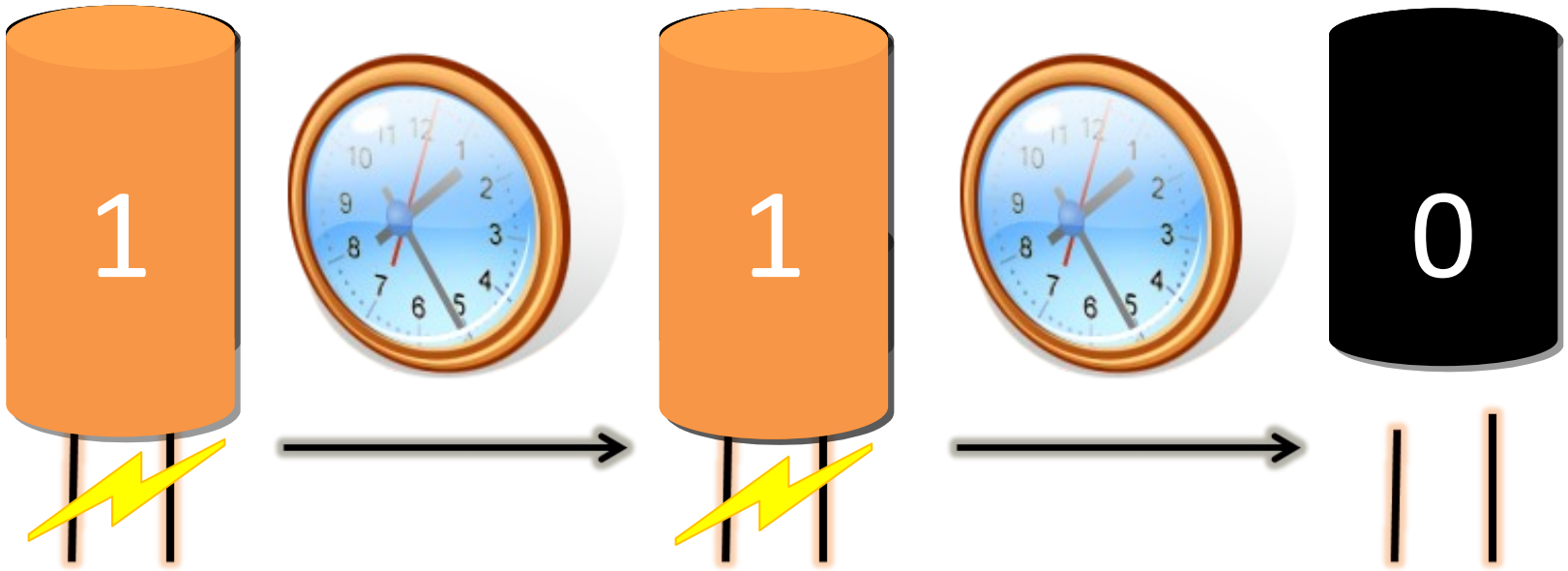
DRAM is an array of tiny capacitors.

To write a bit, the capacitor is charged.

When power is on, the state is refreshed every 10 µs.

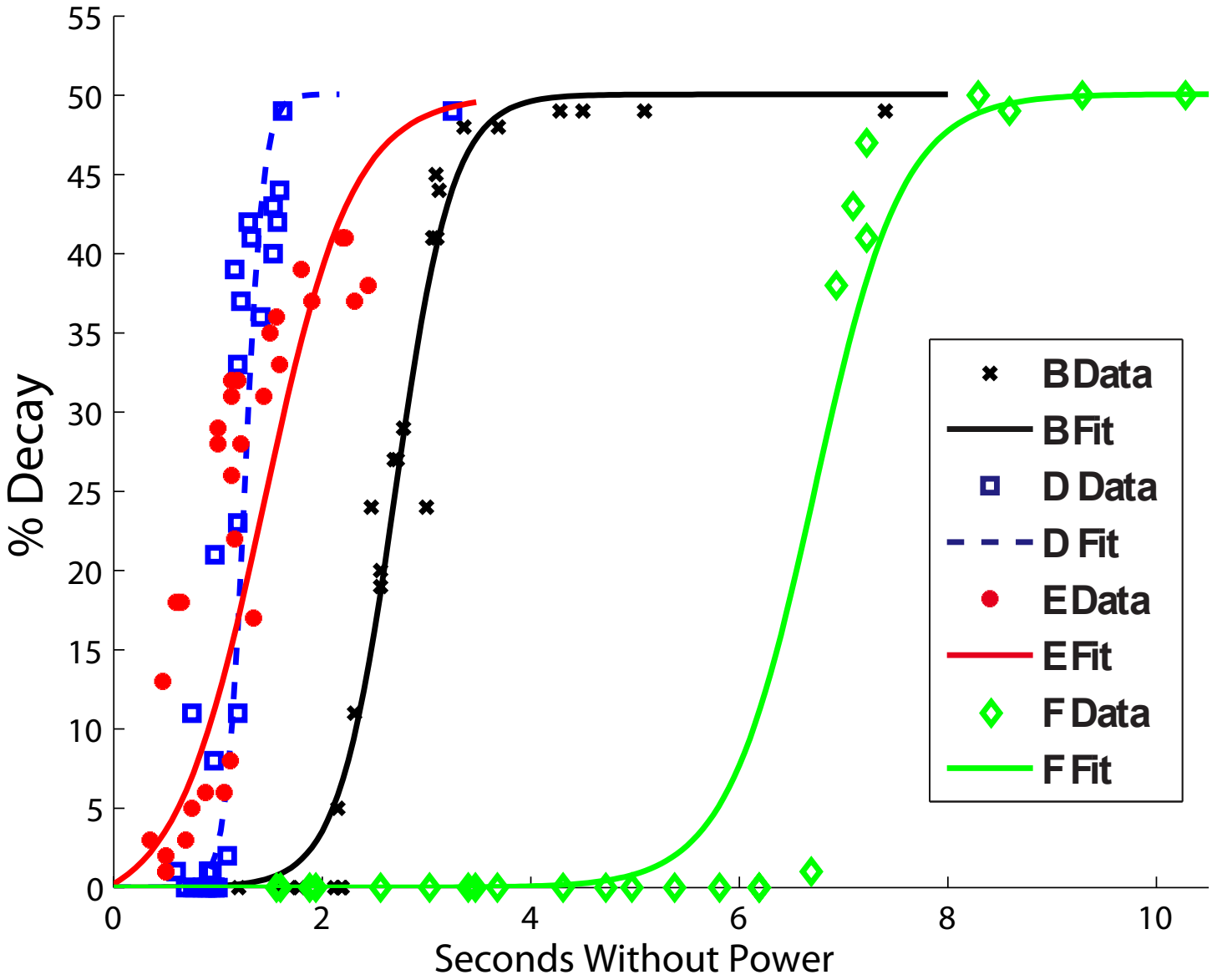
Without power, they discharge to a ground state.

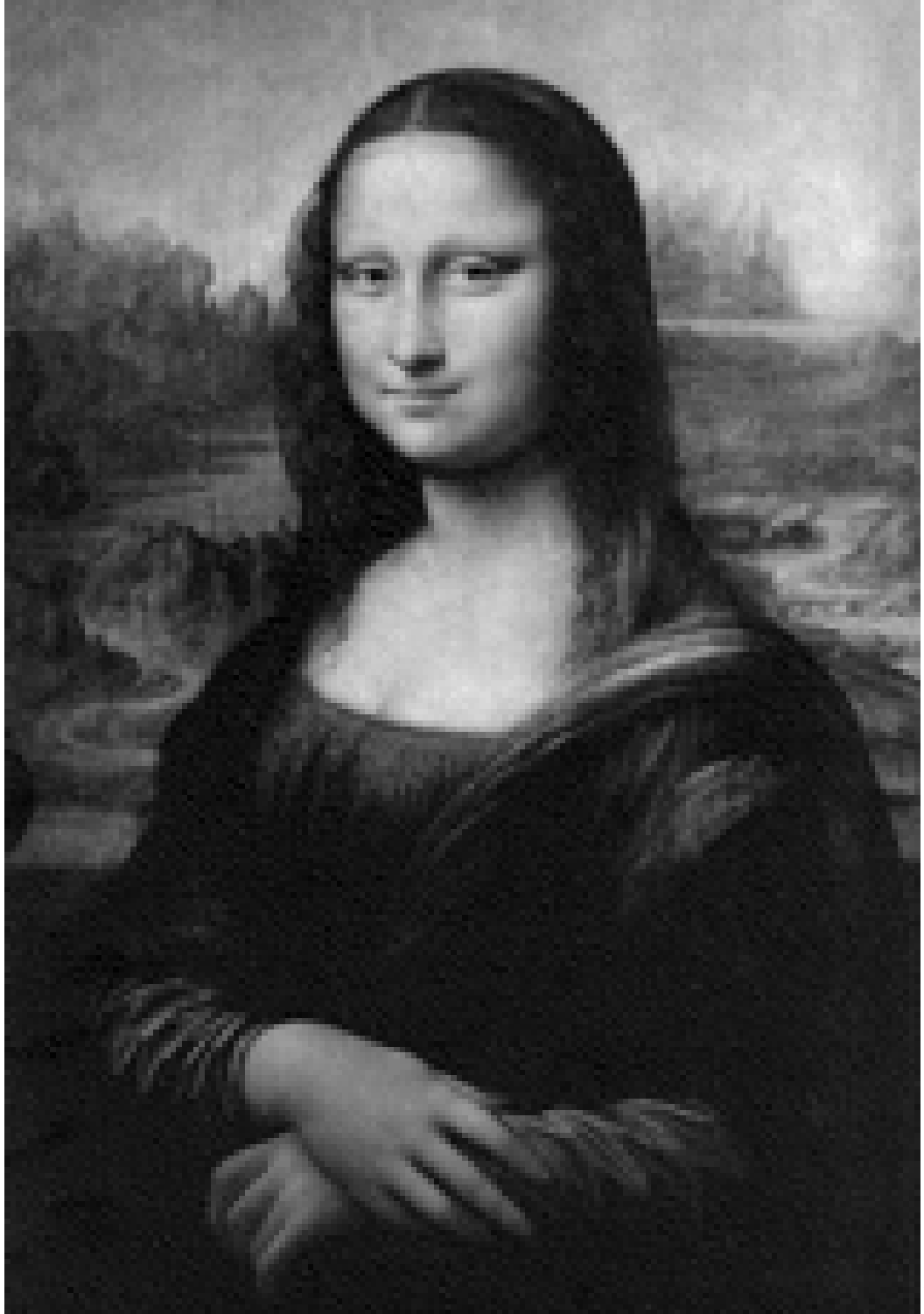




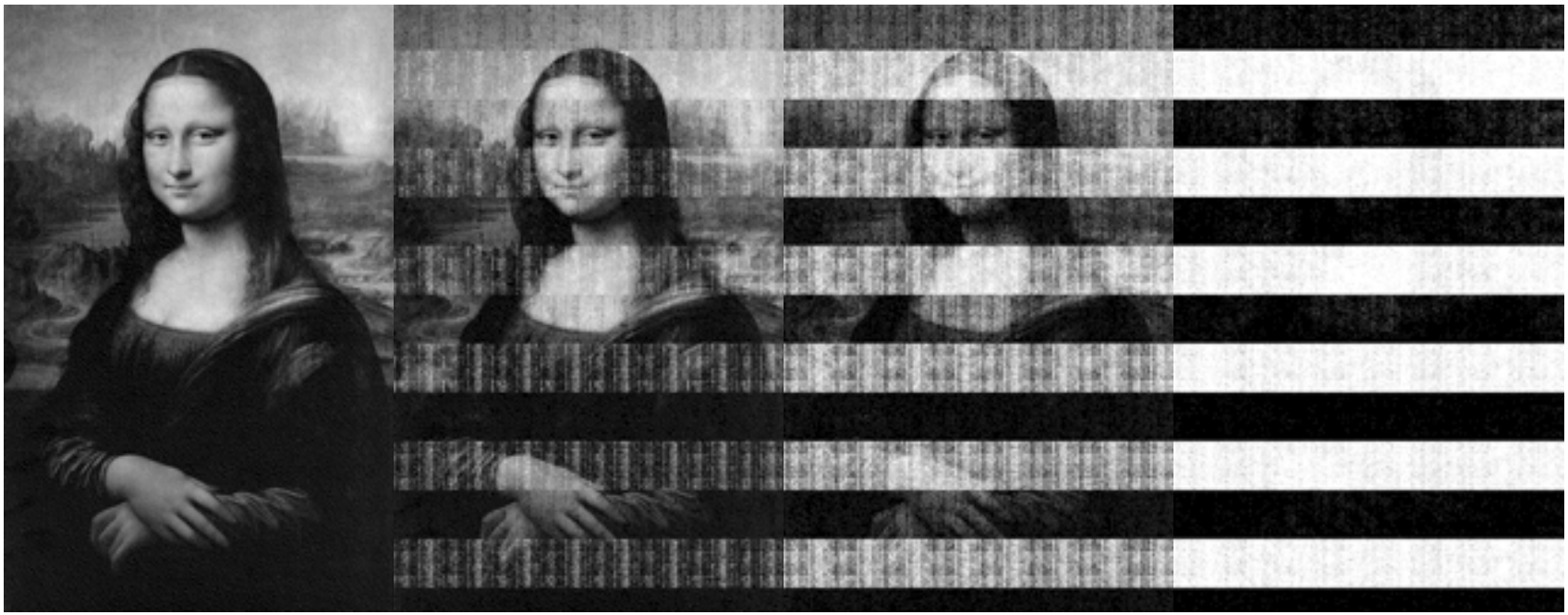
But this process takes seconds to minutes.

DRAM Decay Rates





The Persistence of Memory



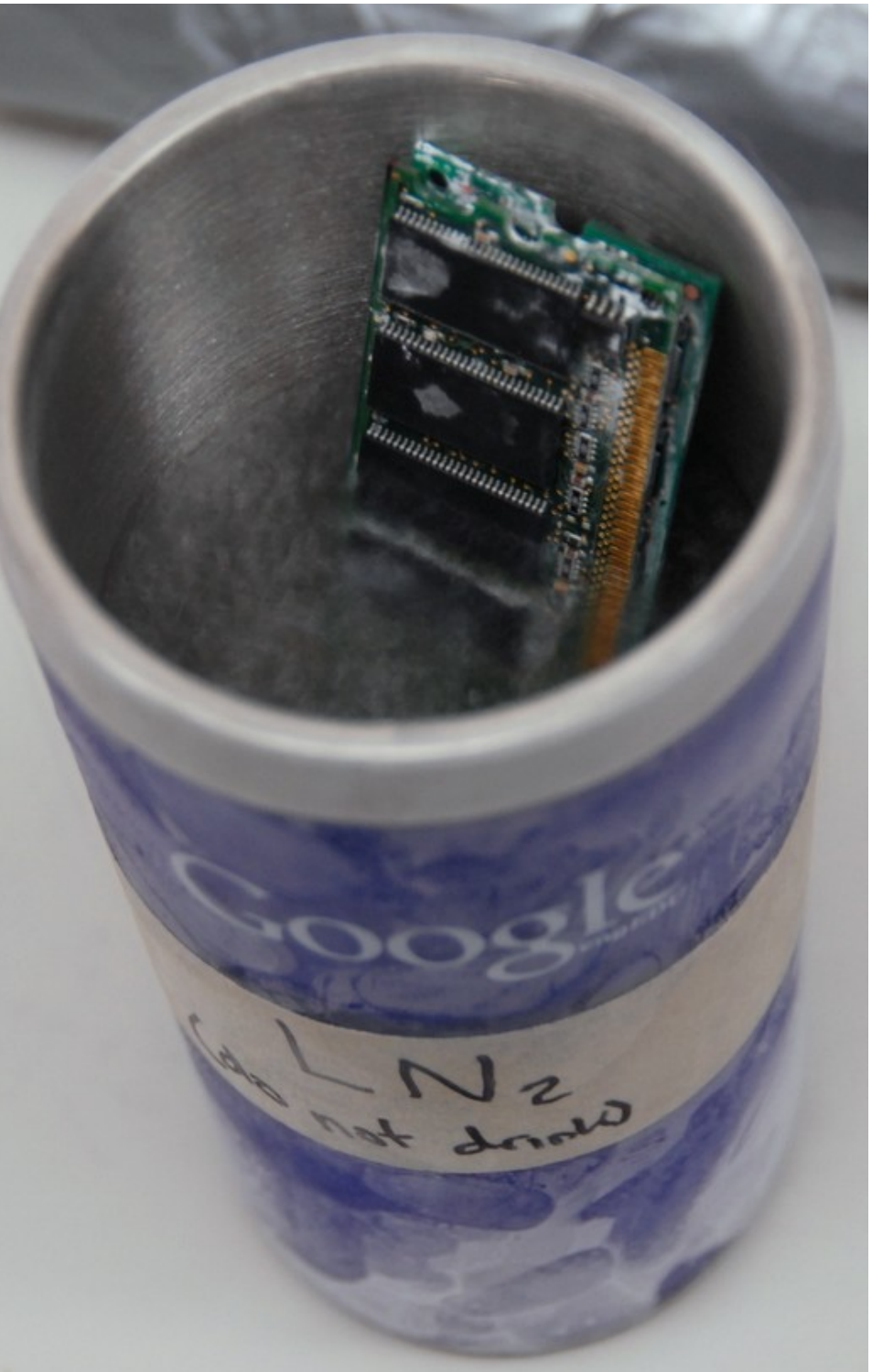
5s. 30s. 1m. 5m.

Slowing Decay by Cooling

-50°C $< 0.2\%$ decay


F5
F6
F7


Esc
F1
F2
F3
Tab
START/STOP
SET
MODE
SPLIT/RESET
SELECT
SUN MON TUE WED THU FRI SAT
6:40
29
GOWalkin
by SPORT



Even cooler

Liquid Nitrogen -196°C

$< 0.1\%$ decay after 1 hour

(not necessary in practice)

Easy targets: Unsanitized data

Plain text passwords from Loginwindow.app in OS X 10.4, 10.5.

Bug had been around for 4 years.

Slightly more difficult targets: Encryption keys

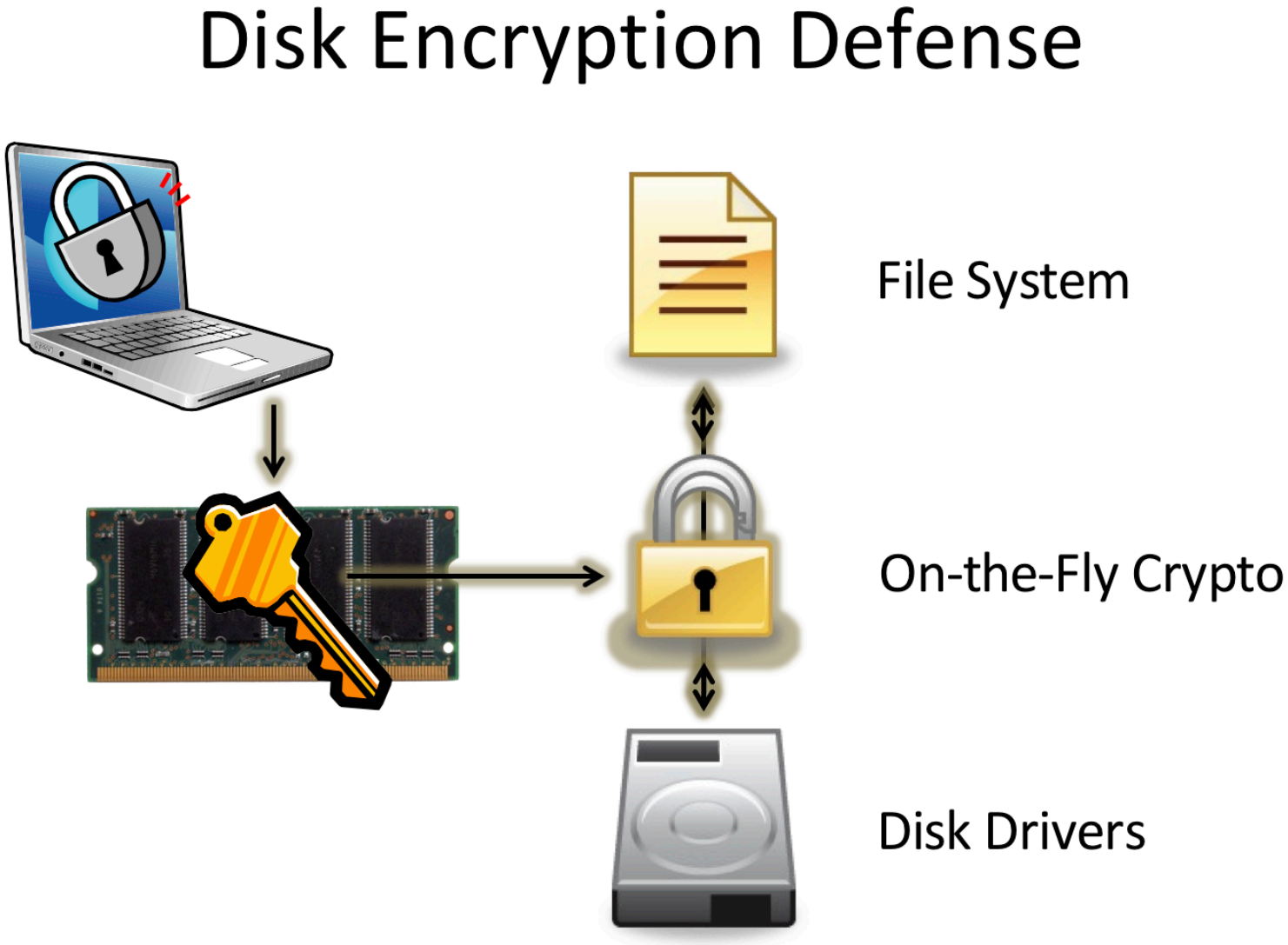
How Safe is a Stolen Laptop?

Unlocked
Unencrypted
No protection

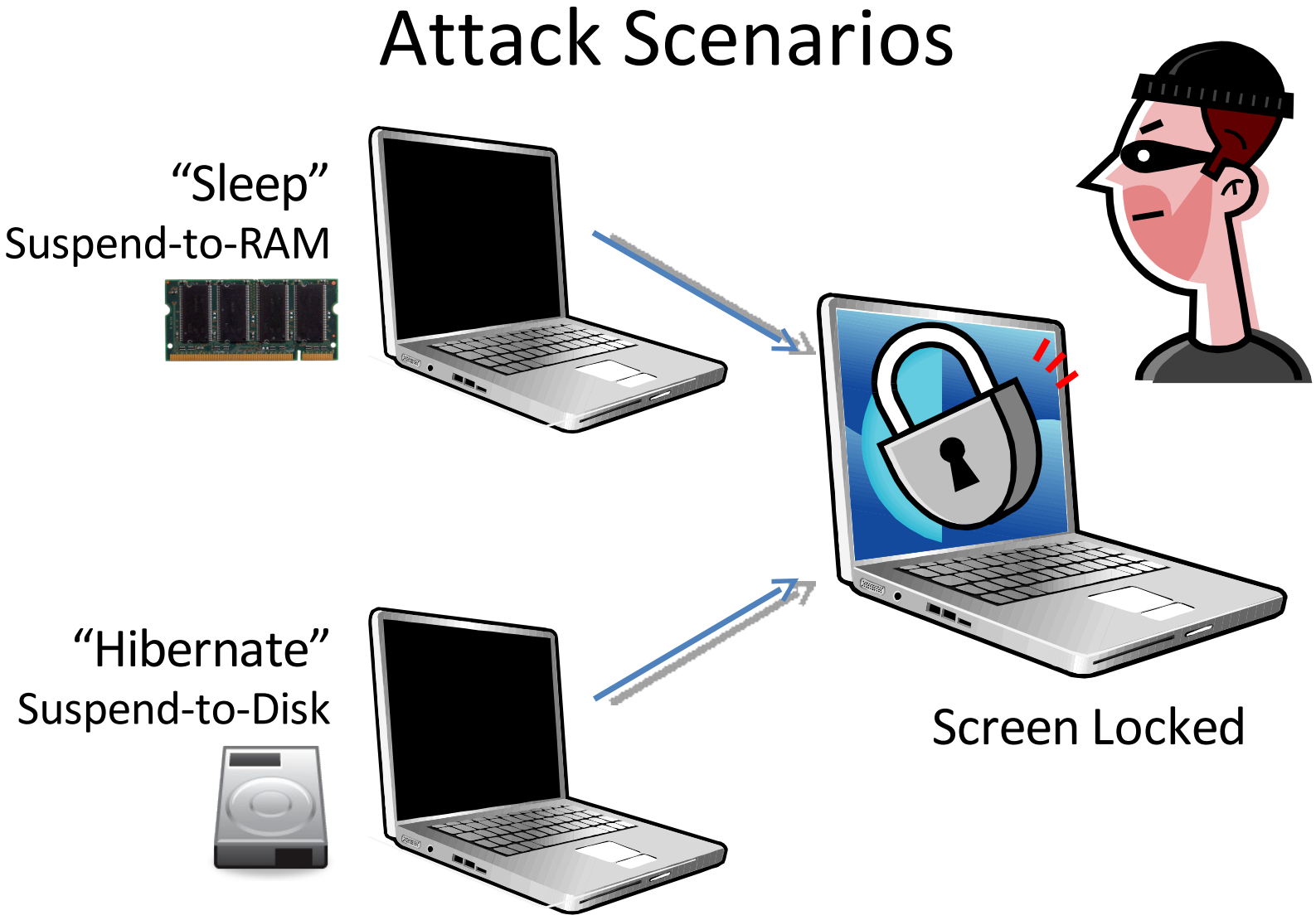
Locked
Unencrypted
Minimal protection

Locked
Encrypted
Industry best practice

Slightly more difficult targets: Encryption keys



Slightly more difficult targets: Encryption keys

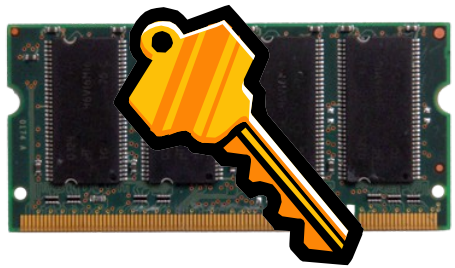


Slightly more difficult targets: Encryption keys

Security Assumptions

The encryption is strong

The OS protects the key in RAM



... the attacker might reboot to circumvent the OS, but since RAM is volatile, the key will be lost...

... Right?

Capturing Residual Data

Residual data can be captured easily, with no special equipment

Complication

Booting a full OS overwrites large areas of RAM.

Solution

Boot a small low-level program to dump contents of memory.

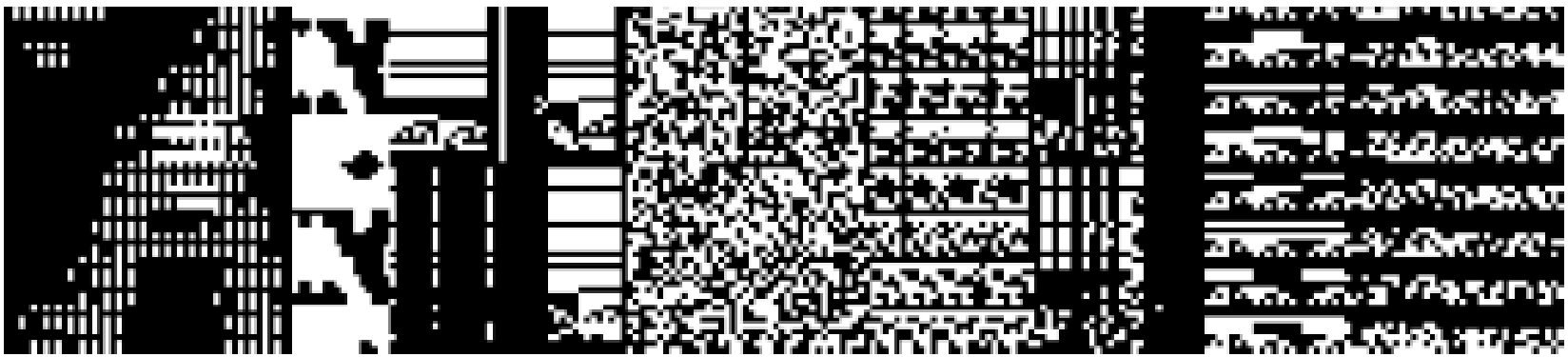
Implementations

PXE Dump (9 KB) EFI Dump (10 KB) USB Dump (22 KB)

Available at `citp.princeton.edu/memory/code/`

Delivering the Attack

Looking for cryptographic keys



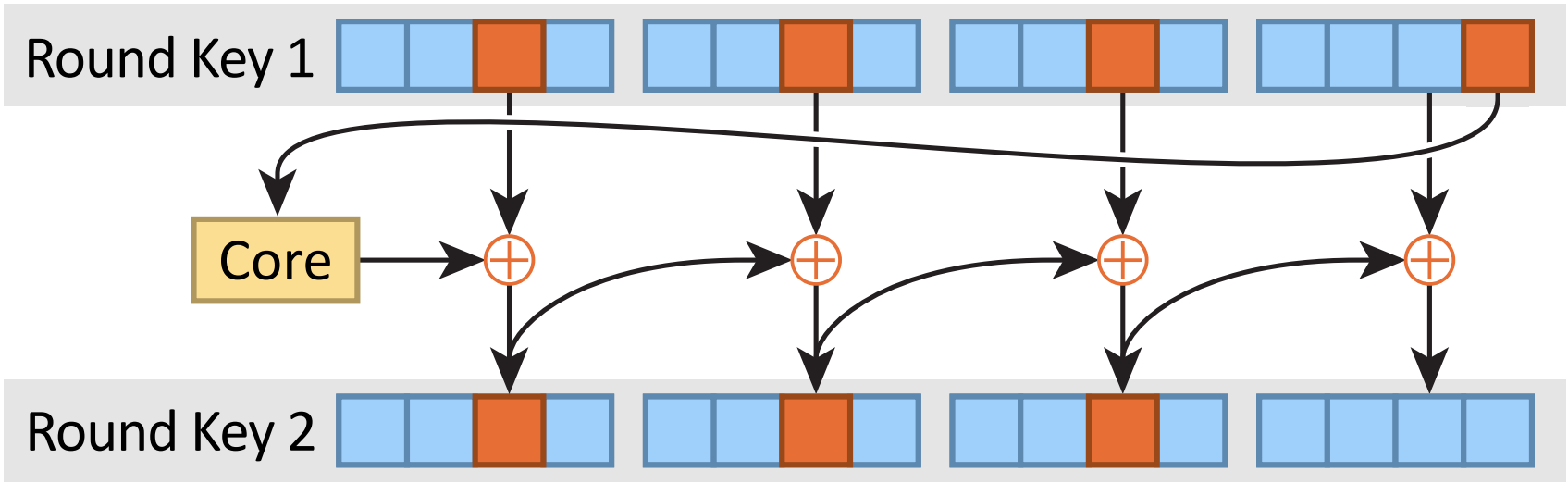
"Playing hide and go seek with stored keys"
Shamir and van Someren

The reality of the entropy approach

```
a92ef19e b2e83d02 51a09daa d0623117 28453d00 0da246f2 5882d33c b0beb65e
b69071fa 4a85f137 ff7caefa f7a019bb a62c79b8 bc7955f3 efc04cb5 4e8c1269
85b56054 c170f9f9 369561ba 677f5d27 36c9a58b d5ddc488 0260cb2f f8f0e332
7c63bbde ee74db08 e580e9e3 cf929c05 ae649dfe 8d1a6e90 182e8799 7e93273c
ae1f7e0d aca6fdbd a6359d03 84b6d124 1c4ccf00 96370107 66a5303e 3cba213e
bbc0f510 641a6f81 0a7ddcc4 51ecbaab d4f6a796 b1cda4f9 d92ba57c eec54d26
718ac1c2 cf8ad02d 9a9562ee 704eea5e bb8d048e 5594b185 93d8fc4d f1e40ed1
ac156c13 05f0d710 8b62c429 4788629f e43467f4 37384584 9bfcac39 2b88cfcf
7aeaa9c9 56bfd90e d85bf171 9e798057 04a9aa5c 5cbeb5a0 0b61f75a 45a2b79d
b9ff57b7 35343877 81dbb9ba 17cc0373 91fde936 8c667375 35a88d0d e26efeae
4a4ac62d 8c21c752 9b47c688 c7be140d 61330f81 22a65c65 aada5ca2 a5d2f34c
e03ce71f 25ab4468 18b6d67b 62d0ce24 436ca259 cdd9e465 db4c8f67 24850d50
8ed617ba 61198ded b821fdf8 b30e485b 6c305a26 7be21618 0c4ab38f 4ea27c25
579ea08b 98e4664e f65e9efd 4f8bfe27 7b7bd863 a0d3e085 9f5ebf45 95181712
8aa2cade eb672c5b 80c57212 09bbbb92 3b9abfce 316372b0 08e15357 2154de36
09a5fca9 e93da242 74989498 8f00527d 25f655ff 617c8522 2a57d285 0fb12c25
cd3903a1 ab4006a7 552d3540 7e67bf8c 78d0838a 73402e8f 7df18dd9 fa861d09
fdd48056 b3cd3db7 e7350adb 5b2dd1c3 9382f77d 1c3f2600 3ef0140e af3af64d
121a145d 55518ae3 b50eae0a 2cd6f0b5 de9b8bd4 61f3501f b83e4ff7 d993262e
fe1dffda b2927243 39df80c3 b3a6093e f799963c bb1f6ba3 339a8e84 d0db7532
ab99c941 7bee0ff7 2e992559 158c6341 f491ad79 7648dd06 5aa3fd18 851e1de8
00ffdb5c 53f07901 7ed48402 5ac66697 6e984007 8429141e 173e9576 00b8986f
5bdc7c4b 0ea28766 ac5bcd7b bdc489bc 4c63f3ab aeab3a01 91b8135b d067ada3
38947871 3c0eb5f3 1dc1f138 623db910 8dac3059 9608c72f 8818d1ed 62863723
104cab65 a261eb72 9ec20c68 cbb4c1c6 48007ffe e239e3a0 e3c9c0fd 0220f689
0eda3387 481d952e b1810a6e 004aeb22 9531d287 9b191215 91c637fa 12855102
a9b389f0 0a2d2e0a 391dd0f0 a8b8826e d784f4d8 31a1ec24 7449c188 ba310424
```

Finding Keys: Use the structure of the key data.

AES implementations typically precompute a sequence of round keys from the single 128 or 256-bit key.



Generation of the 128-bit AES key schedule.

Identifying AES keys in memory

Every encryption program we looked at computed and stored the key schedules in exactly the same way.

```
c3284c9f ed58820e c1e923df 78a30623
e59446f1 08ccc4ff c925e720 b186e103
9e5c020b 9690c6f4 5fb521d4 ee33c0d7
9074c1b5 06e40741 59512695 b762e642
bcdd6b33 ba396c72 e3684ae7 540aaca5
bafd0cb2 00c460c0 e3ac2a27 b7a68682
a95428d6 a9904816 4a3c6231 fd9ae4b3
c40090ff 6d90d8e9 27acbad8 da365e6b
bb579527 d6c74dce f16bf716 2b5da97d
44a6d9ef 92619421 630a6337 4857ca4a
92f482ad 0095168c 639f75bb 2bc8bff1
```

Identifying AES keys in memory

To identify an AES key schedule in memory, scan for a block of memory that has the properties of a key schedule.

```
c3284c9f ed58820e c1e923df 78a30623
e59446f1 08ccc4ff c925e720 b186e103
9e5c020b 9690c6f4 5fb521d4 ee33c0d7
9074c1b5 06e40741 59512695 b762e642
bcdd6b33 ba396c72 e3684ae7 540aaca5
bafd0cb2 00c460c0 e3ac2a27 b7a68682
a95428d6 a9904816 4a3c6231 fd9ae4b3
c40090ff 6d90d8e9 27acbad8 da365e6b
bb579527 d6c74dce f16bf716 2b5da97d
44a6d9ef 92619421 630a6337 4857ca4a
92f482ad 0095168c 639f75bb 2bc8bff1
```

Identifying AES keys in memory

To identify an AES key schedule in memory, scan for a block of memory that has the properties of a key schedule.

```
c3284c9f ed58820e c1e923df 78a30623
e59446f1 08ccc4ff c925e720 b186e103
9e5c020b 9690c6f4 5fb521d4 ee33c0d7
9074c1b5 06e40741 59512695 b762e642
bcdd6b33 ba396c72 e3684ae7 540aaca5
bafd0cb2 00c460c0 e3ac2a27 b7a68682
a95428d6 a9904816 4a3c6231 fd9ae4b3
c40090ff 6d90d8e9 27acbad8 da365e6b
bb579527 d6c74dce f16bf716 2b5da97d
44a6d9ef 92619421 630a6337 4857ca4a
92f482ad 0095168c 639f75bb 2bc8bff1
```

Identifying AES keys in memory

To identify an AES key schedule in memory, scan for a block of memory that has the properties of a key schedule.

```
c3284c9f ed58820e c1e923df 78a30623
e59446f1 08ccc4ff c925e720 b186e103
9e5c020b 9690c6f4 5fb521d4 ee33c0d7
9074c1b5 06e40741 59512695 b762e642
bcdd6b33 ba396c72 e3684ae7 540aaca5
bafd0cb2 00c460c0 e3ac2a27 b7a68682
a95428d6 a9904816 4a3c6231 fd9ae4b3
c40090ff 6d90d8e9 27acbad8 da365e6b
bb579527 d6c74dce f16bf716 2b5da97d
44a6d9ef 92619421 630a6337 4857ca4a
92f482ad 0095168c 639f75bb 2bc8bff1
```

Identifying AES keys in memory

To identify an AES key schedule in memory, scan for a block of memory that has the properties of a key schedule.

```
c3284c9f ed58820e c1e923df 78a30623
e59446f1 08ccc4ff c925e720 b186e103
9e5c020b 9690c6f4 5fb521d4 ee33c0d7
9074c1b5 06e40741 59512695 b762e642
bcdd6b33 ba396c72 e3684ae7 540aaca5
bafd0cb2 00c460c0 e3ac2a27 b7a68682
a95428d6 a9904816 4a3c6231 fd9ae4b3
c40090ff 6d90d8e9 27acbad8 da365e6b
bb579527 d6c74dce f16bf716 2b5da97d
44a6d9ef 92619421 630a6337 4857ca4a
92f482ad 0095168c 639f75bb 2bc8bff1
```

How to identify RSA keys in memory?

Try multiplying blocks of memory together?

Try decrypting with every block of memory?

PKCS #1: RSA Cryptography Standard

```
RSAPublicKey ::= SEQUENCE {
    modulus             INTEGER,  -- n
    publicExponent      INTEGER   -- e
}

RSAPrivateKey ::= SEQUENCE {
    version             Version,
    modulus             INTEGER,  -- n
    publicExponent      INTEGER,  -- e
    privateExponent     INTEGER,  -- d
    prime1              INTEGER,  -- p
    prime2              INTEGER,  -- q
    exponent1           INTEGER,  -- d mod (p-1)
    exponent2           INTEGER,  -- d mod (q-1)
    coefficient         INTEGER,  -- (inverse of q) mod p
    otherPrimeInfos     OtherPrimeInfos OPTIONAL
}
```

PKCS #1: BER-encoding

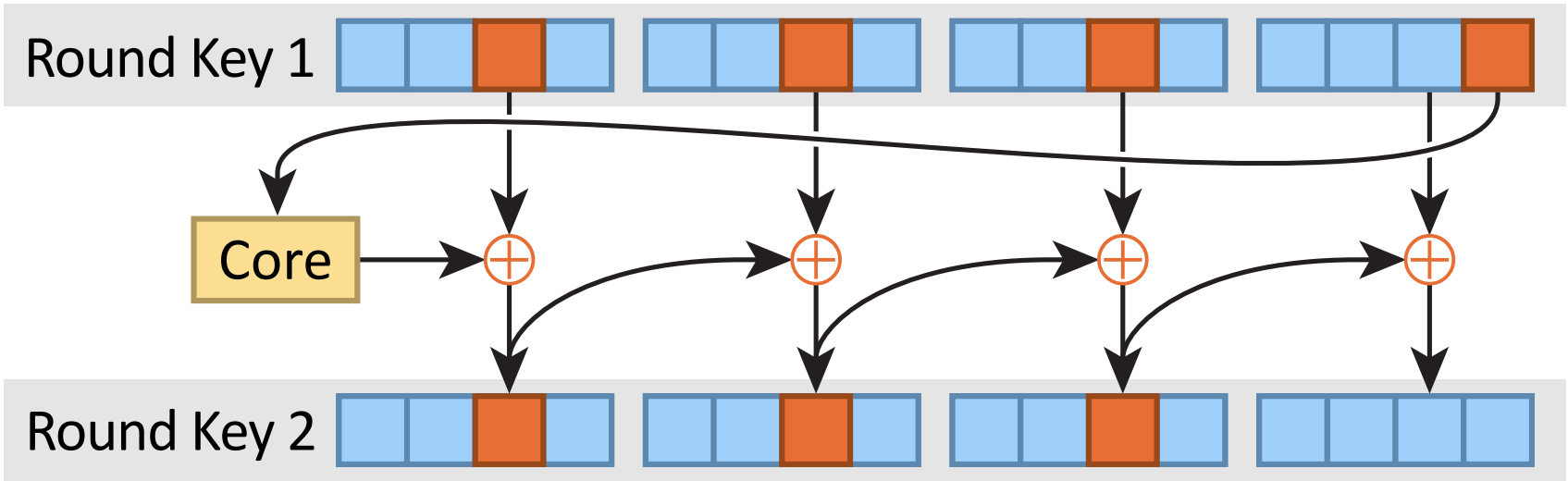
```
30
82 02 5d
    02
    01
        00
    02
    81 81
        00 99 63 66 d5 ...
    02
    03
        01 00 01
    02
    81 80
        6e 77 ef 54 a7 ...
    ...
```

```
SEQUENCE
length:  605 bytes
    INTEGER
    length:  1 byte
        (version)
    INTEGER
    length:  129 bytes
        (n)
    INTEGER
    length:  3 bytes
        (e)
    INTEGER
    length:  128 bytes
        (d)
    ...
```

What if the recovered memory contains bit errors?

Correcting Errors in Cryptographic Keys: AES

Use the structure of redundant key data to correct errors.



Can retrieve an AES key from 30% of a key schedule in seconds.

Correcting Errors in Cryptographic Keys: RSA

Use the structure of redundant key data to correct errors.

$$pq = N$$

$$ed = 1 \pmod{(p-1)(q-1)}$$

$$ed_p = 1 \pmod{p-1}$$

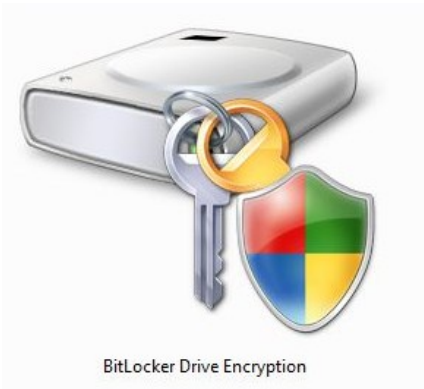
$$ed_q = 1 \pmod{q-1}$$

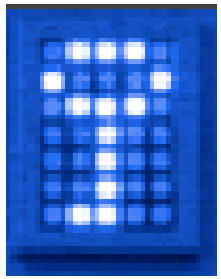
Can retrieve an RSA key from 27% of key data in seconds.

Attacking disk encryption systems

1. Cut the power to the computer.
2. Reboot into a small memory extracting program.
3. Dump the data from RAM to a device of your choosing.
4. Find keys, fix any errors, decrypt hard drive.

Works against:

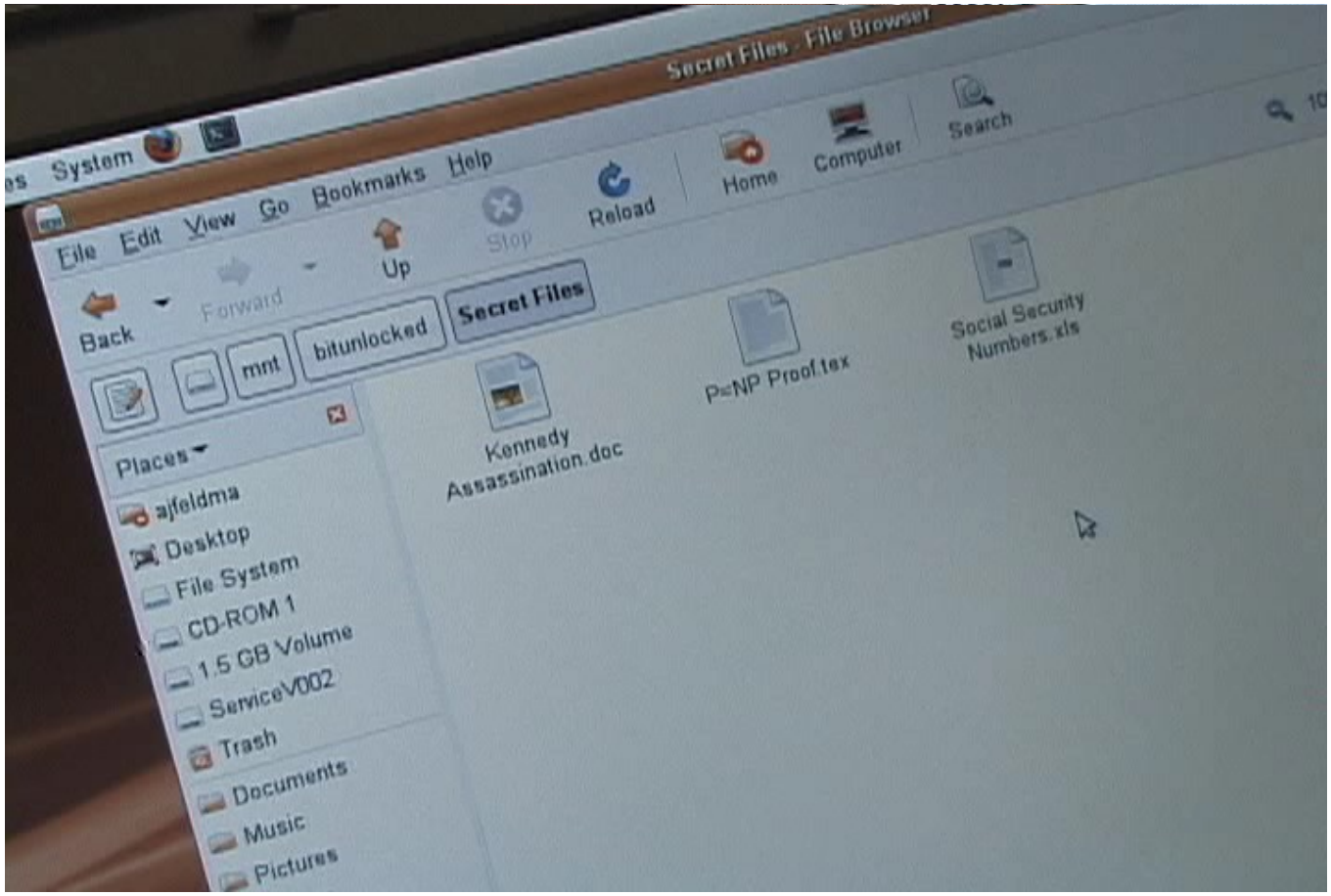




and others...

Microsoft BitLocker Apple Filevault TrueCrypt Loop-AES dm-crypt

"BitLocker, meet Bit*Un*Locker."



Demonstration of fully automated attack:

Connect USB drive, reboot, and browse files

Countermeasures

No Magic Bullet

Possible Mitigations

- Encrypt key in memory when screen-locked.
- Avoid precomputation.
- Fully-encrypted memory
- Trusted Platform Module (TPM)

Encrypt Memory During Sleep

When entering screen-lock/hibernate/sleep:

- Encrypt RAM with user's password

When awakened:

- Require user's password to decrypt RAM

Minor behavioral changes

Zzzz...

Avoid Precomputation

This makes recovery harder, but:

- Hurts Performance
  Having key schedule speeds computation
- Attacker can reduce the incidence of errors
- Alternative recovery schemes may perform well without this data

Encrypted Memory

Memory encrypted with key

- Randomly chosen at boot

On cache read/write

- Data encrypted/decrypted when written/read

CPU reset clears key

TPM?

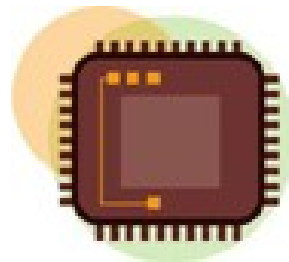
Current TPMs may help attacker

Windows BitLocker in Basic Mode

- On boot, OS loads key from TPM into RAM
  (No password)
- Vulnerable to cold-boot attack
  (Even if completely off)

Future TPMs may help

- Need bulk encryption



Since the original paper...

... a lot of interesting follow-up work.

- Reviving an entire computer, VPN sessions and all.
  "Bootjacker: compromising computers using forced restarts."
- Improvements in key error-correction.
- Theoretical cryptographic work for "leakage-resilient" cryptosystems.
  "Cryptography without (hardly any) secrets"

For video, paper, and source code, visit:

`citp.princeton.edu/memory`