Summary of:

RSA Key Extraction via Low-Bandwidth Acoustic Cryptanalysis*

Daniel Genkin
Technion and Tel Aviv University
danielg3@cs.technion.ac.il

Adi Shamir
Weizmann Institute of Science
adi.shamir@weizmann.ac.il

Eran Tromer
Tel Aviv University
tromer@cs.tau.ac.il

December 18, 2013

http://www.cs.tau.ac.il/~tromer/acoustic/

Credit (including pictures and algorithms) to authors of the paper ars technica

MAIN MENU MY STORIES: 25 FORUMS SUBSCRIB

RISK ASSESSMENT / SECURITY

New attack steals e-mail decry capturing computer sounds

Scientists use smartphone to extract secret key of nearby PC ru

by Dan Goodin - Dec 19 2013, 6:25am ICT

NBC NEWS TECHNOLOGY

Gadgets Security Internet Innovation Video games

The sound of secrets: New hacking technique infiltrates by hearing — or touch

Devin Coldewey, NBC News

Dec. 19, 2013 at 4:08 PM ET

HACKING PRIVACY

Researchers crack the world's toughest encryption by listening to the tiny sounds made by your computer's CPU

By Sebastian Anthony on December 18, 2013 at 2:27 pm | Comment

Forbes New Posts +7 posts this hour Most Popular Year's Hottest Startups Lists Best Actors For The

7 Stocks to Sell for 2014

Tim Worstall, Contributor

I write about business and technology.

TECH | 12/21/2013 @ 9:22AM | 10,185 views

Researchers Break RSA 4096 Encryption With Just A Microph And A Couple Of Emails

The Register®

Data Centre Software Networks Security Policy Business Jobs Hardware Science Bootnotes

SECURITY

Code-busters lift RSA keys simply by listenin to the noises a computer makes

Don't put your mobe down by your machine. In fact just chuck it i the river

By John Leyden, 19th December 2013

RSA

RSA

- Key generation:

Choose two large primes, p and q, and calculate $n = pq$

Select e relatively prime with $\phi(n)$, calculate d as inverese of e

$$\mathrm{PU} = (e, n)$$

$$\mathrm{PR} = (d, n)$$

- Encryption of message:

$$C = M^e \bmod n$$

- Decryption of ciphertext:

$$M = C^d \bmod n$$

RSA 4096-bit

- RSA supports different “key” lengths: 1024, 2048, 4096 bits
- Key generation:
  - p is 2048 bits, q is 2048 bits
  - $n = pq$ is 4096 bits
  - e often 65,537 (16 bits)
  - d is calculated; about same length as n, ~ 4000 bits
- Decryption/Signing, i.e. using private key, M, $C < n$:

$$C^d \bmod n$$

$$(\textit{very large number})^{(\textit{very large number})} \bmod n$$

RSA Implementation

- Split the modular exponentiation of 4096-bit number into two modular exponentiations of 2048-bit numbers
  - Chinese Remainder Theorem
  - $d_p = d \pmod{p\text{-}1}$
  - $d_q = d \pmod{q\text{-}1}$
  - $q_{inv} = q^{-1} \pmod{p}$
- Decryption/Signing:
  - $m_p = C^{d_p} \bmod p$
  - $m_q = C^{d_q} \bmod q$
  - $h = q_{inv}\,(m_p - m_q) \pmod{p}$
  - $M = m_q + hq$

Two steps using smaller exponents; Increases speed by factor of 4 compared to one step with large exponent

History

- 1978: Ron Rivest, Adi Shamir and Len Adlemen

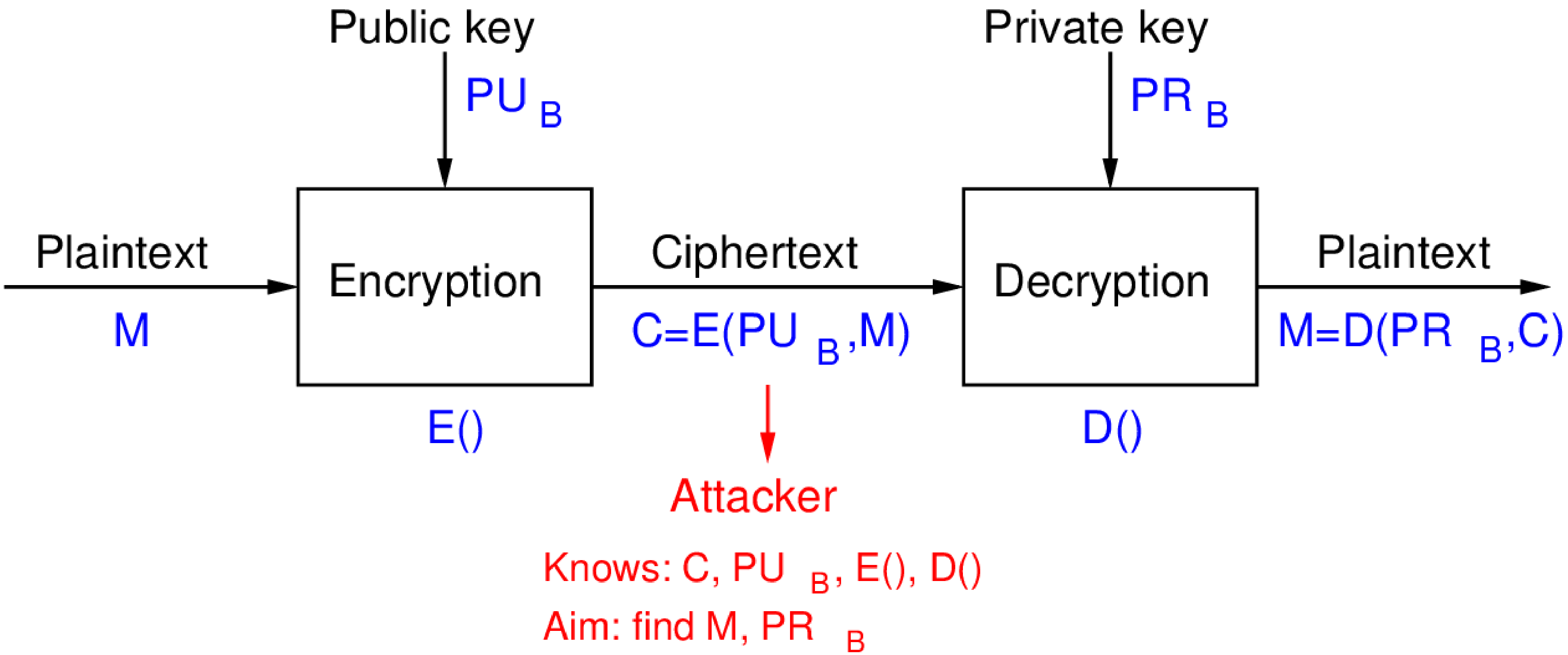
algorithm

company

- 1982: Formed company - RSA Security
  - Sells authentication tokens and BSAFE library of cryptographic operations (alternative to OpenSSL)
- 1995: Employees created digital certificate company (VeriSign)
- 2006: Acquired by EMC
- 2013: Alleged NSA backdoor in random number generator proposed and used by RSA

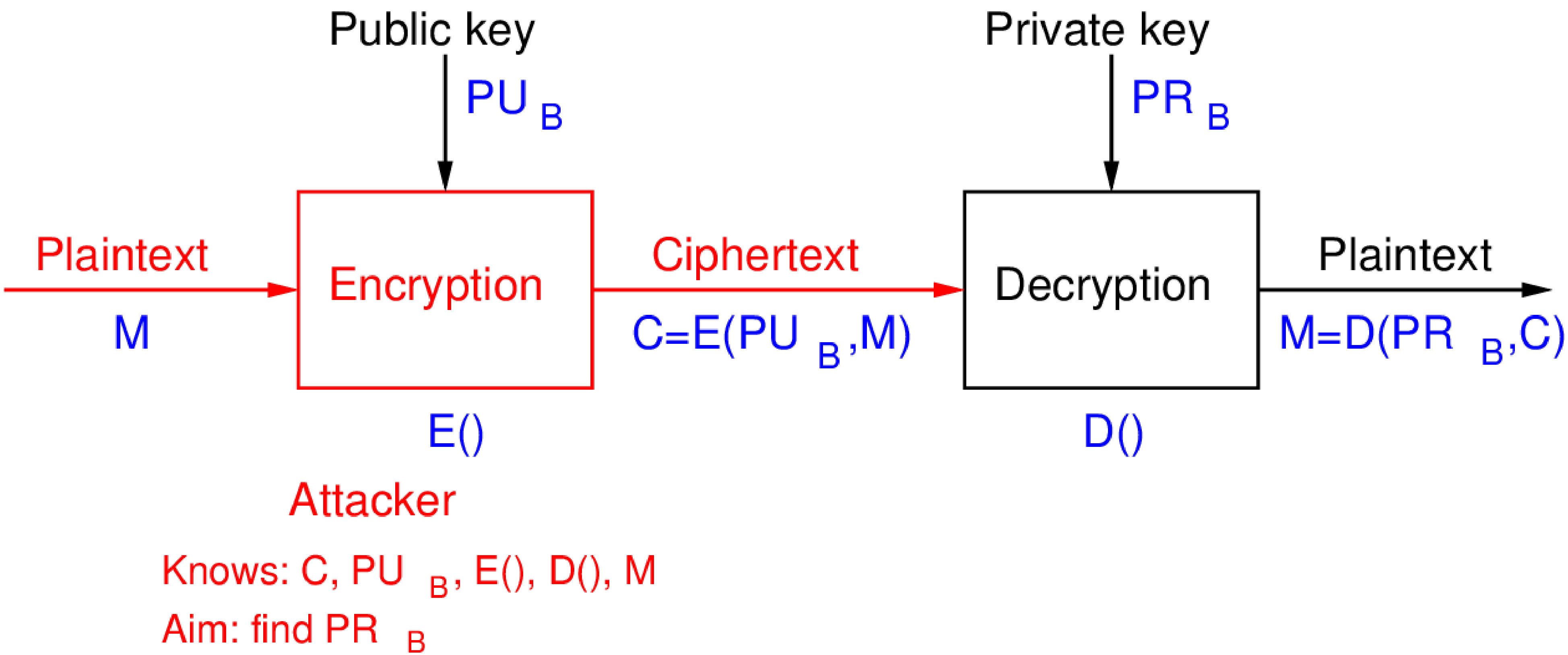
Side Channel Attacks

Ciphertext Only Attacks



Attack intercepts ciphertext, aims to find the plaintext and/or private key

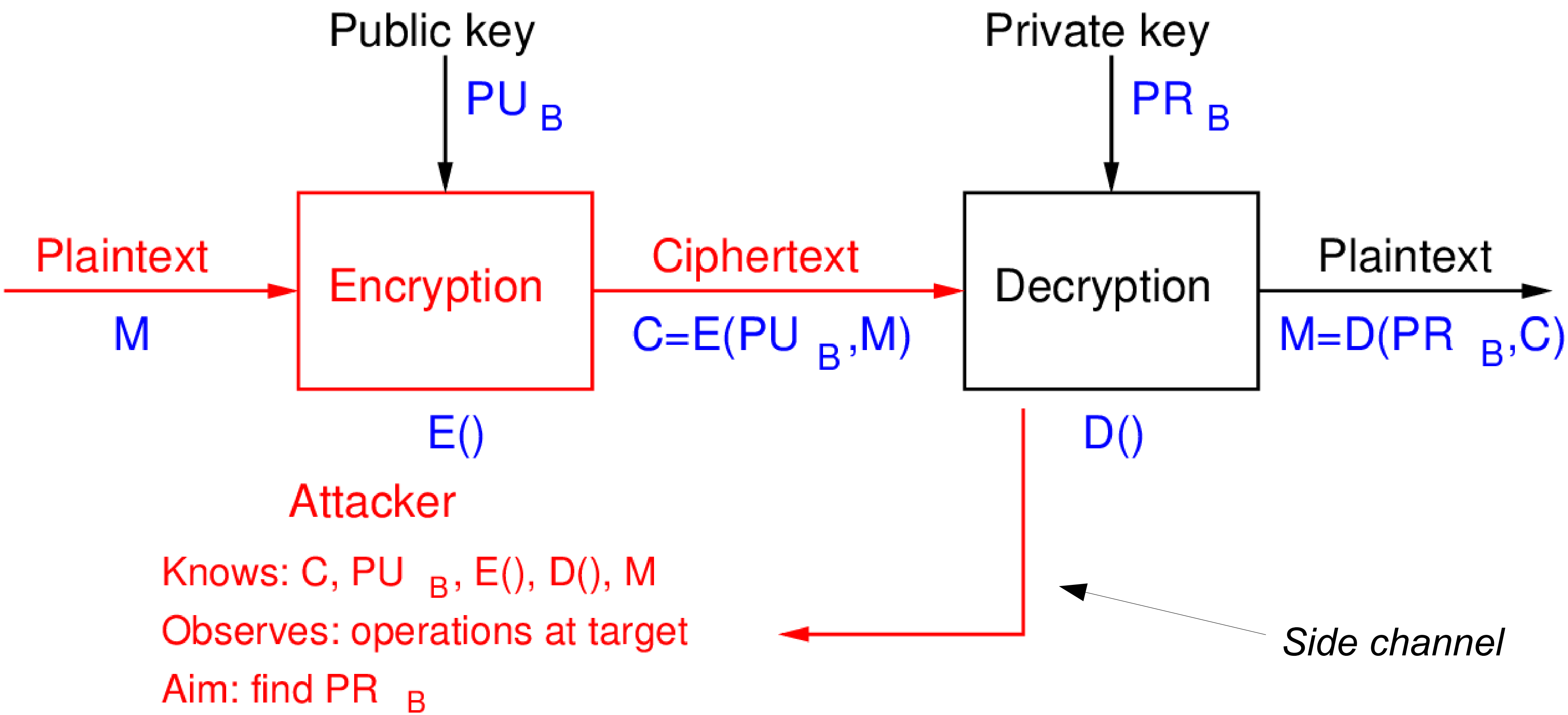
Chosen Plaintext/Ciphertext Attacks



Attacker can choose multiple ciphertext (and plaintext) values and convince target to decrypt them

Aims to find the private key

Side Channel Attack



Attacker can choose multiple ciphertext (and plaintext) values and convince target to decrypt them

Attacker can also observe activities of targets computer

Aims to find the private key

RSA Key Extraction via Low-Bandwidth Acoustic Cryptanalysis

Daniel Genkin
Technion and Tel Aviv University

Adi Shamir
Weizmann Institute of Science

Eran Tromer
Tel Aviv University

December 18, 2013

http://www.cs.tau.ac.il/~tromer/acoustic/
http://www.tau.ac.il/~tromer/papers/acoustic-20131218.pdf

The Attack

1. Send a specially crafted ciphertext to target
2. Record the audio generated by target computer while it is decrypting ciphertext
   - Need recording equipment nearby
   - Different values of q require different operations in decryption, producing different sounds by target
   - Identifying the different sounds allows for determining bits of q
3. Repeat with different ciphertexts until all bits of q are determined
4. Calculate p and d
5. Profit!!!

The Attack

1. Send a specially crafted ciphertext to target
2. Record the audio generated by target computer while it is decryp
    - Nee
    - Diffe proc
    - Iden
3. Repea determ
4. Calcul
5. Profit!

Example

Target runs an email client that automatically decrypts emails. Email client decrypts using targets Private key (d).

Attacker creates the necessary chosen ciphertext and emails to target.

Attacker can repeatedly send emails, making them look like spam. Target email client automatically decrypts and then discards. User doesn't notice.

POSSIBLE

The Attack

1. Send a specially crafted ciphertext to target
2. Record the audio generated by target computer while it is decrypting ciphertext
   - Need recording equipment nearby
   - Different values of q require different operations in decryption, producing different sounds by target
   - Identifying the different sounds allows for determining bits of q
3. Repeat with different … determined
4. Calculate p and d
5. Profit!!!

We will look at this in depth next.

POSSIBLE (with some conditions)

The Attack

1. Send a specially crafted ciphertext to target
2. Record the audio generated by target computer while it is decrypting ciphertext
    - Need recording equipment nearby
    - Different values of q require different operations in decryption, producing different sounds by target
    - Identifying the different sounds allows for determining bits of q
3. Repeat with different ciphertexts until all bits of q are determined
4. Calculate p and d
5. Profit!!!

As described in step 1.

POSSIBLE

The Attack

1. Send a specially crafted ciphertext to target
2. Record the audio generated by target computer while it is decrypting ciphertext
   - Need recording equipment nearby
   - Different values of q require different operations in decryption, producing different sounds by target
   - Identifying the differe … of q
3. Repeat with different … determined
4. Calculate p and d
5. Profit!!!

Public values: e, n, C, M

If you also know q:
$n = pq$ therefore $q = n/p$
$\phi(n) = (p-1)(q-1)$
Calculate d (same as key generation)

EASY

Listening to a computer

- CPUs change their power consumption depending what they need to do
  - Depends on type and number of operations, e.g. MUL, ADD
- Leads to vibrations of electrical components in power supply circuitry
- Vibrations create sound (acoustic emanations)
- So what?

  If we can listen to the sound and, if we can distinguish what operations are being performed while decrypting, and if the operations depend on specific private keys, then can learn the private key

A lot of ifs ...

If we can listen to the sound and, if we can distinguish what operations are being performed while decrypting, and if the operations depend on specific private keys, then can learn the private key

- Microphones pickup frequencies from up to 20kHz, even up to 100kHz (with lower sensitivity). Sound from CPU activity differs in frequencies than other sources (fan, hard disk etc)
- Different operations produce acoustic signals (sound) with different spectrograms
- Creating chosen ciphertexts trigger different operations in RSA decryption (modular exponentiation) depending on key

How to record sound of target computer?

Experimental Setup: Fixed

Figure 6: Parabolic microphone (same as in Figure 5), attached to the portable measurement setup (in a padded briefcase), attacking a target laptop from a distance of 4 meters. Full key extraction is possible in this configuration and distance (see Section 5.4).

Experimental Setup: Portable

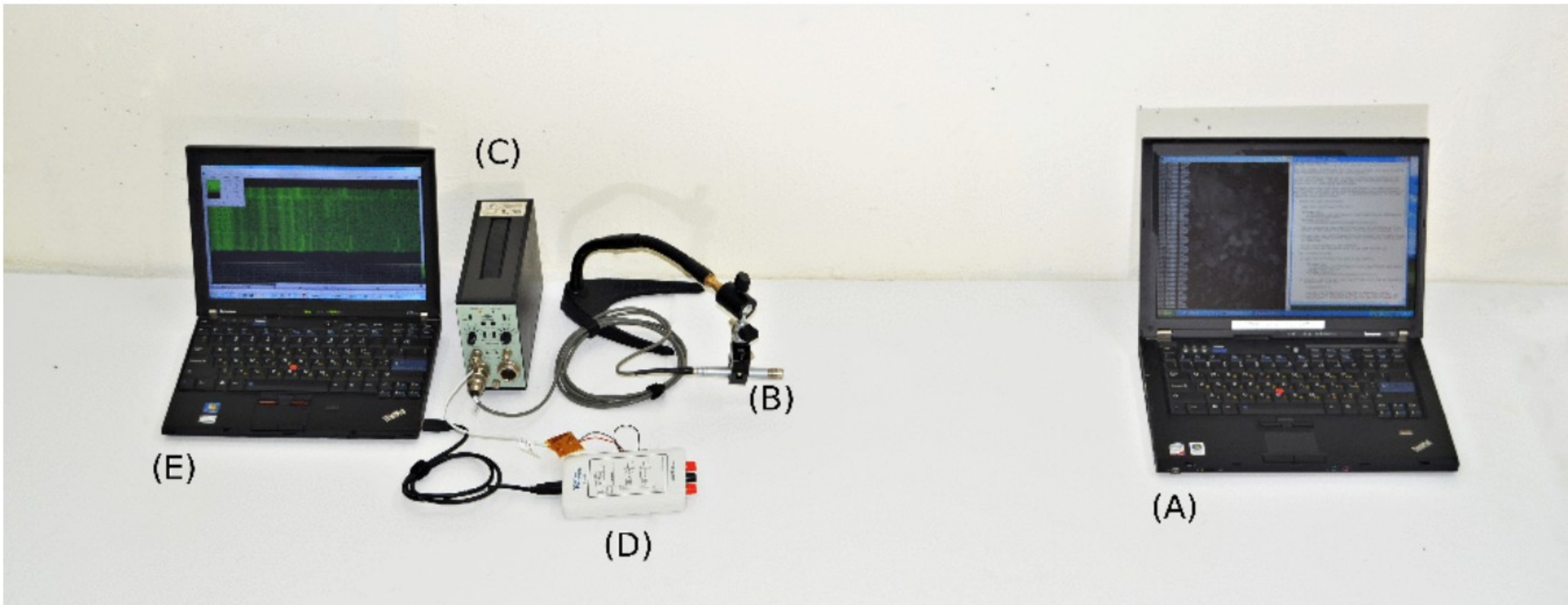


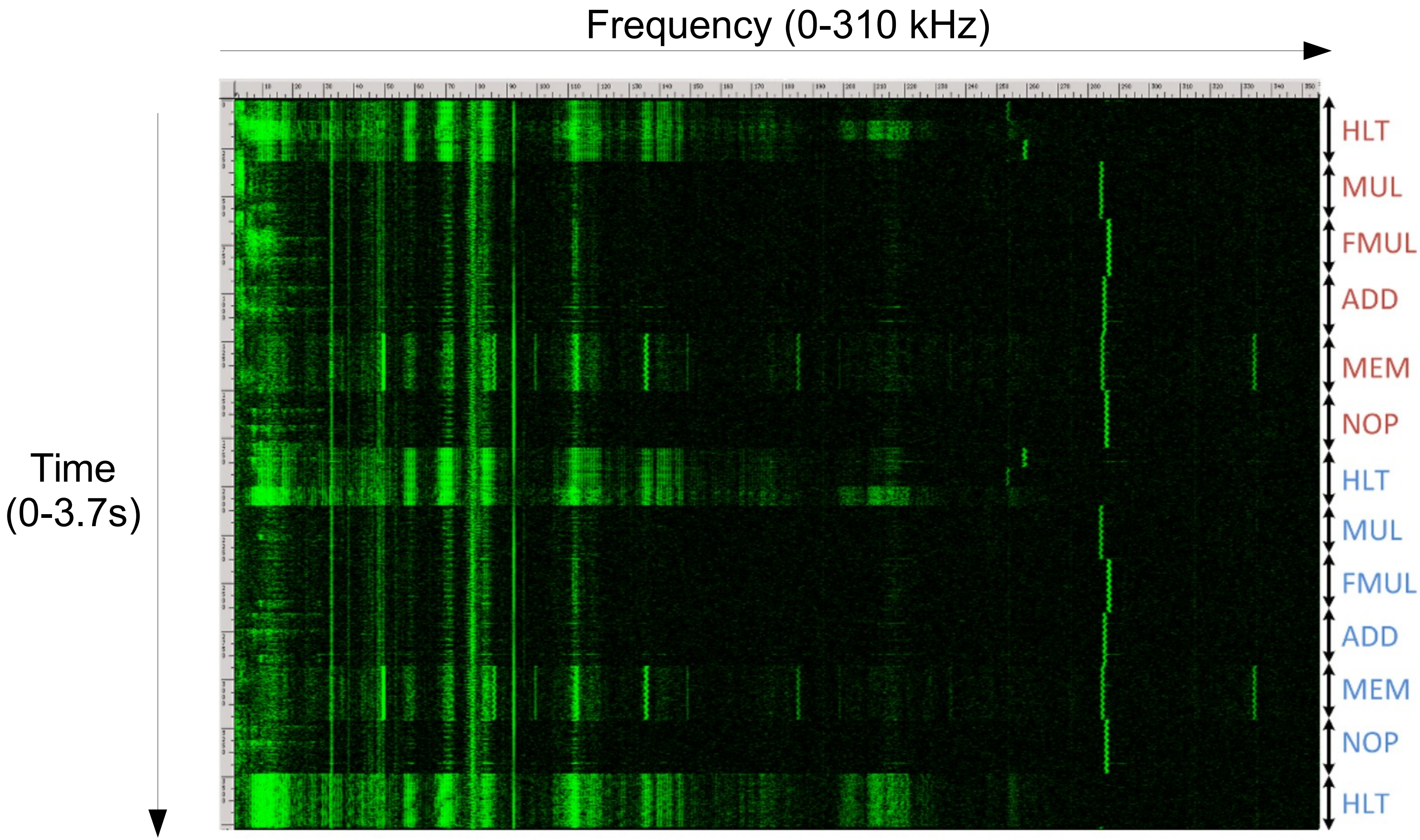
Figure 3: Photograph of our portable setup. In this photograph (A) is a Lenovo ThinkPad T61 target, (B) is a Brüel&Kjær 4190 microphone capsule mounted on a Brüel&Kjær 2669 preamplifier held by a flexible arm, (C) is a Brüel&Kjær 5935 microphone power supply and amplifier, (D) is a National Instruments MyDAQ device with a 10 kHz RC low-pass filter cascaded with a 150 kHz RC high-pass filter on its A2D input, and (E) is a laptop computer performing the attack. Full key extraction is possible in this configuration, from a distance of 1 meter (see Section 5.4).

Experimental Setup: Mobile

Figure 4: Physical setup of a key recovery attack. A mobile phone (Samsung Note II) is placed 30 cm from a target laptop. The phone's internal microphone points towards the laptop's fan vents. Full key extraction is possible in this configuration and distance (see Section 5.4).

Can different CPU operations be detected by sound?

Frequency Spectrogram of CPU Operations



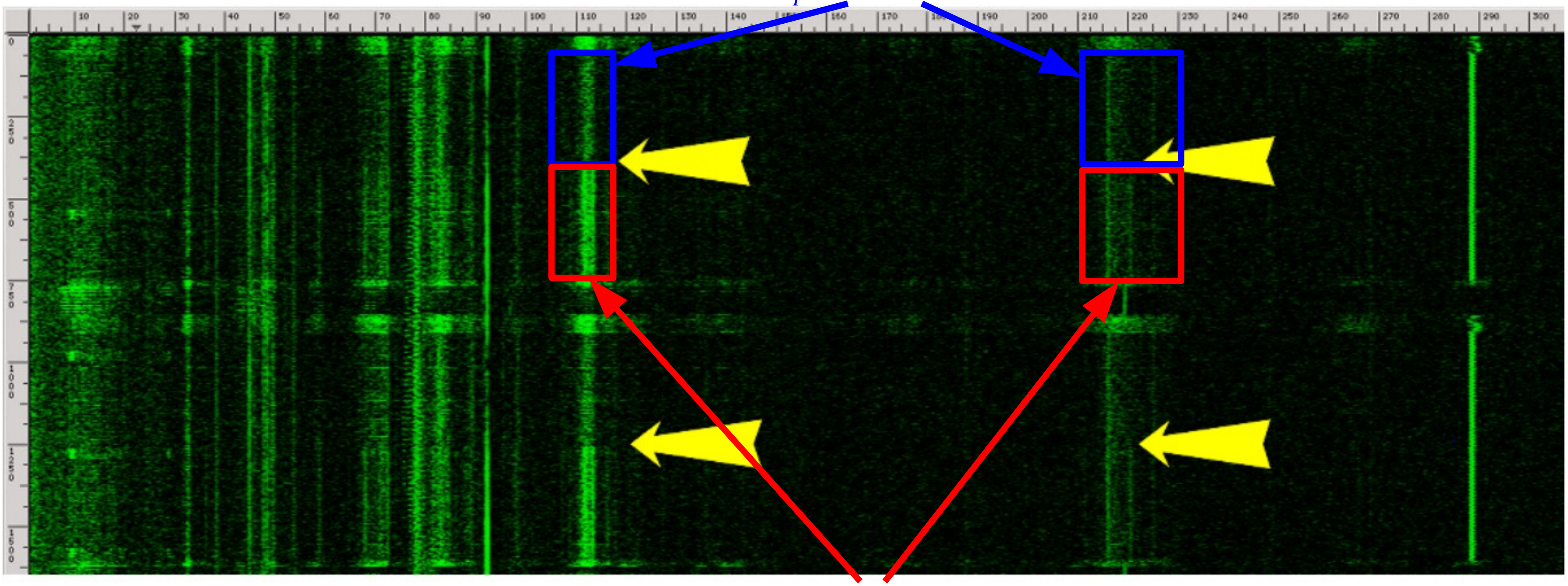
“Greener” the value, larger the signal magnitude

Figure 7: Acoustic measurement frequency spectrogram of a recording of different CPU operations using the Brüel&Kjær 4939 microphone capsule. The horizontal axis is frequency (0–310 kHz), the vertical axis is time (3.7 sec), and intensity is proportional to the instantaneous energy in that frequency band.

mod *p* and mod *q* can be distinguished

Yellow arrows show where RSA changes from mod *p* to mod *q* modular exponentiation

$m_p = c^{d_p} \bmod p$



$m_q = c^{d_q} \bmod q$

Figure 12: Acoustic signature (1.6 sec, 0–300 kHz) of two GnuPG RSA signatures executed on the Evo N200. The recoding was made using the lab-grade setup and the Brüel&Kjær 4939 high-frequency microphone capsule. The transitions between p and q are marked with yellow arrows.

Another laptop, Freq up to 40kHz

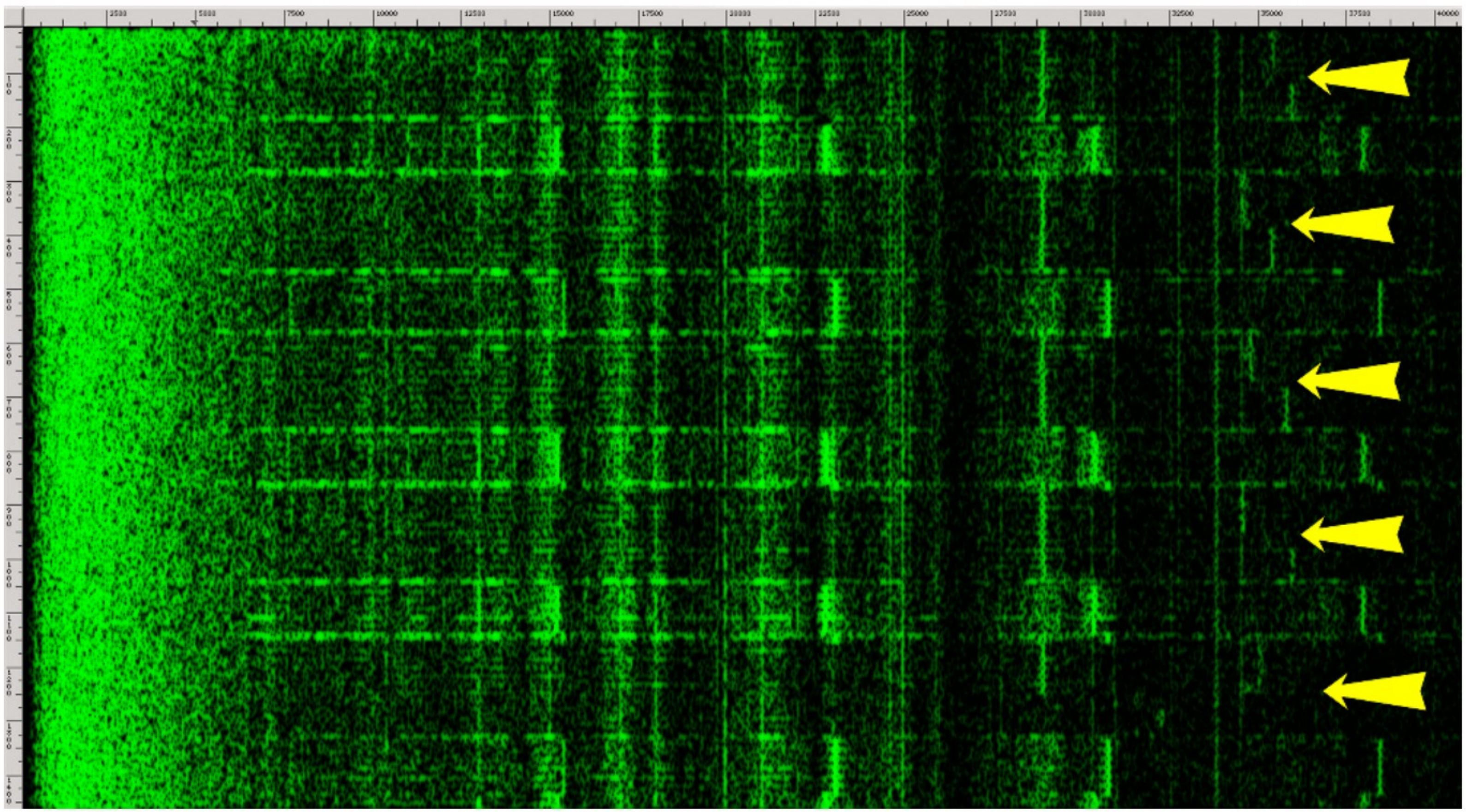


Figure 15: Acoustic signature (1.4 sec, 0–40 kHz) of five GnuPG RSA signatures executed on a Lenovo ThinkPad T61. The recoding was made using the lab-grade setup and the Brüel&Kjær 4190 microphone capsule. The transitions between p and q are marked with yellow arrows.

Are the CPU operations dependent on the private key?

(and if so, can we detect the different operations?)

Approach

- Choose a ciphertext such that the decryption by the target will require different operations depending on the target's key
  - "Target's key" is q in this attack
- Focus on a single bit in q at a time
- Attacker wants the decryption to sound different depending on that bit of q
  - Send a chosen ciphertext to target
  - If attacker can detect the different sounds, then can detect that bit of q
- Repeat by sending different chosen ciphertexts to detect subsequent bits of q
  - Either repeat for all 2048 bits of q
  - Or use Coppersmith attack: require about 1024 bits of q

Modular Exponentiation Algorithm

Algorithm 1 GnuPG's modular exponentiation (see function mpi_powm in mpi/mpi-pow.c).

Input: Three integers c, d and q in binary representation such that $d = d_n \cdots d_1$.

m: m_q

d: d_q (2048 bits)

q (2048 bits)

Output: $m = c^d \mod q$.

procedure MODULAR_EXPONENTIATION(c, d, q)
- **if** SIZE_IN_LIMBS(c) > SIZE_IN_LIMBS(q) **then**
  - $c \leftarrow c \mod q$ — Reduce ciphertext *c* if greater than *q*
- $m \leftarrow 1$
- **for** $i \leftarrow n$ **downto** 1 **do** — Loop 2048 times
  - $m \leftarrow m^2$
  - **if** SIZE_IN_LIMBS(m) > SIZE_IN_LIMBS(q) **then**
    - $m \leftarrow m \mod q$
  - **if** SIZE_IN_LIMBS(c) < KARATSUBA_THRESHOLD **then** ▷ defined as 16
    - $t \leftarrow$ MUL_BASECASE(m, c) ▷ Compute $t \leftarrow m \cdot c$ using Algorithm 3
  - **else** — Multiply current *m* and ciphertext *c*
    - $t \leftarrow$ MUL(m, c) ▷ Compute $t \leftarrow m \cdot c$ using Algorithm 5
  - **if** SIZE_IN_LIMBS(t) > SIZE_IN_LIMBS(q) **then**
    - $t \leftarrow t \mod q$
  - **if** $d_i = 1$ **then**
    - $m \leftarrow t$
- **return** m

end procedure

q Modular Exponentiation (Simplified)

MODULAR_EXPONENTIATION (c, d, q) {

  c = c mod q ← Reduce ciphertext *c*

  $m_q = 1$

  for i = 2048 .. 1 {

    $m_q = m_q^{\,2}$

    …

    $t = m_q * c$

    …

  }

  return m_q

}

2048 multiplications of *c* and *m*

Choosing the Ciphertext

- q is 2048 number

  $q_{2048}q_{2047}q_{2046} \cdots q_3q_2q_1$

- Assume we know the first $(i - 1)$ bits of q
  - E.g. $i = 4$, we know: $q_{2048}q_{2047}q_{2046} = 110$
- Aim: find the next bit of q
  - E.g. q_{2045}: is it 0 or 1?
- Create ciphertext with first $(i - 1)$ bits of q, then 0, then all 1's

  $q_{2048}q_{2047}q_{2046}011111...11111$

- Send chosen ciphertext to target for decryption

q Modular Exponentiation of Chosen Ciphertext

```
MODULAR_EXPONENTATION (c, d, q) {
    c = c mod q
    m_q = 1
    for i =
        m_q = m
        …
        t = m_q
        …
    }
    return m_q
}
```

$c = q_{2048}q_{2047}q_{2046}0 \quad 11111\ldots11111$

$q = q_{2048}q_{2047}q_{2046}q_{2045}q_{2044}q_{2043}\ldots$

If $q_{2045} = 1$, $c < q$:

$c \bmod q = c$ c doesn't change; still 2048 bits with many 1's at right

If $q_{2045} = 0$, $c \geq q$:

$c \bmod q = ?$ c changes; smaller, random looking number

q Modular Exponentiation of Chosen Ciphertext

```
MODULAR_EXPONENTATION (c, d, q) {
   c = c mod q
   m_q = 1
   for i = 2048 .. 1 {
      m_q = m_q^2
      …
      t = m_q * c
      …
   }
   return m_q
}
```

If q_{2045} = 1, c < q:

c doesn't change; still 2048 bits with many 1's at right

2048 multiplications with structured, 2048 bit c

If q_{2045} = 0, c ≥ q:

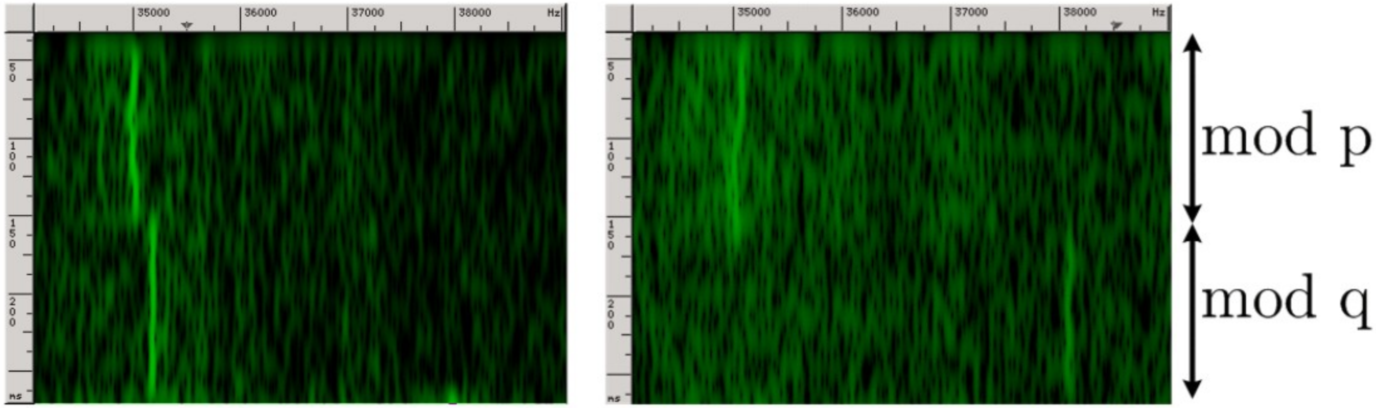
c changes; smaller, random looking number

2048 multiplications with random, shorter c

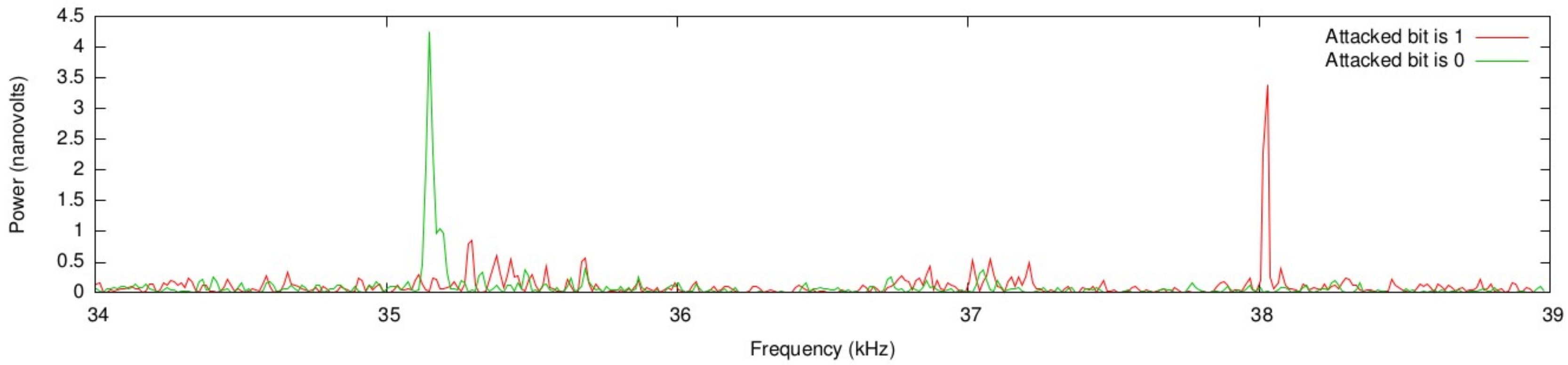
Hope

- If $q_{2045} = 1$
  - Loop of 2048 multiplications with 2048-bit c that is structured (all 1's on right)
- If $q_{2045} = 0$
  - Loop of 2048 multiplications with shorter (less than 2048-bit) c that is random looking
- Hope that the implementation of the loops will require different CPU operations
- Hope that the difference of CPU operations will be detectable when listening to the acoustic emanations (sound from computer)
  - If so, then by detecting different sounds can determine if q_{2045} is 0 or 1
- Once attacker knows q_{2045}, then repeat for q_{2044} and so on
  - (Note q_{2048} is typically 1, to ensure q is large)

Frequencies change depending on bit of *q*



(a) attacked bit is zero (b) attacked bit is one



(c) Frequency spectra of the second modular exponentiation

Figure 16: Acoustic emanations of RSA decryption for various values of the attacked bit ($q_{2039} = 1$ and $q_{2038} = 0$).

Profit!!!

Is the attack realistic?

Conditions of the Attack

- Target computer:
  - RSA Implementation: GnuPG (up to version 1.4.15, Oct 2013)
  - Enigmail Thunderbird plugin for OpenPGP encrypted emails
  - Specific laptops
- Authors expect similar attacks will be successful for other software, protocols and hardware
  - Give example of distinguishing ElGamal keys

Example Attack Scenarios

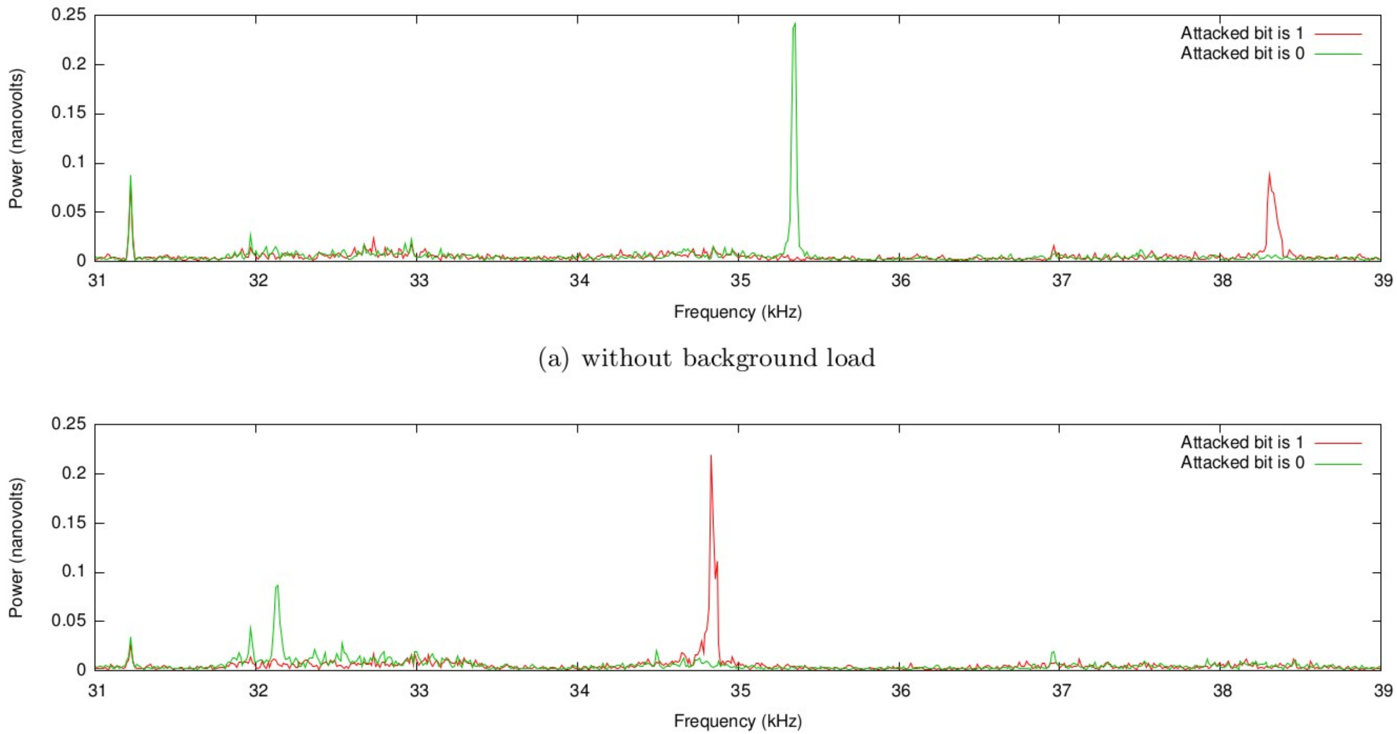
- App on mobile phone, leave it near target computer
- Compromise target's mobile phone; automatic attack initiation and reporting to remote server
- Compromised target computer listens to itself
- “Bugs”:
  - Leave small dedicated devices near target
  - Place in laptop lock cables, charging stations, presentation podiums
- Include inside server case, placed in co-location facility; listen to other servers

Can the attack be prevented?

Acoustic Shielding?

- Add material that weakens acoustic signal
- Increase in target computer cost
- Hard to shield fan vent holes in laptops

Use CPU for other operations at same time?



(a) without background load

(b) with background load

Figure 28: Acoustic measurement frequency spectra of the second modular exponentiation with and without constant load.

Other CPU operations will not necessarily hide the decryption operations

Listen to music?

- Create some other noise while decrypting
- Music and other common sounds have different spectrum (up to 20kHz) than CPU operations (around 35 kHz)
- Would need a special acoustic noise generator designed to create noise that hides CPU operations

Ciphertext randomization?

- Before decrypting, perform an operation on the ciphertext (similar to encryption)
  - Produces random output, *r*
  - Decrypt *r*
  - Apply inverse operation to the real plaintext
- Works
  - But requires extra processing
- Similar approach:
  - Randomize modulus *n* during modular exponentation

GnuPG is being updated to incorporate fixes

Go read the FAQ and paper

http://www.cs.tau.ac.il/~tromer/acoustic/