Get Your Hands Off My Laptop: Physical Side-Channel Key-Extraction Attacks on PCs

(extended version)

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

Itamar Pipman
Tel Aviv University
itamarpi@tau.ac.il

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

July 31, 2014

Abstract

We demonstrate physical side-channel attacks on a popular software implementation of RSA and ElGamal, running on laptop computers. Our attacks use novel side channels, based on the observation that the "ground" electric potential, in many computers, fluctuates in a computation-dependent way. An attacker can measure this signal by touching exposed metal on the computer's chassis with a plain wire, or even with a bare hand. The signal can also be measured at the remote end of Ethernet, VGA or USB cables.

Through suitable cryptanalysis and signal processing, we have extracted 4096-bit RSA keys and 3072-bit ElGamal keys from laptops, via each of these channels, as well as via power analysis and electromagnetic probing. Despite the GHz-scale clock rate of the laptops and numerous noise sources, the full attacks require a few seconds of measurements using Medium Frequency signals (around 2 MHz), or one hour using Low Frequency signals (up to 40 kHz).

Contents

1 Introduction

1.1 Background

Side-channel attacks that exploit unintentional, abstraction-defying information leakage from physical computing devices have proven effective in breaking the security of numerous cryptographic implementations [And08, MOP07, KJJR11]. However, most research attention has been focused on small devices: smartcards, RFID tags, FPGAs, microcontrollers, and simple embedded devices. The "PC" class of devices (commodity laptop/desktop/server computers) has been studied from the perspective of side channels measured by resident software (see [Hu92] and subsequent works) and from peripherals (e.g., [Kuh03]).

PCs, however, have received little academic attention with regard to physical emanations from cryptographic computations, presumably due to three main barriers. First, PCs have highly complicated system architecture and CPU micro architecture, with many noise sources and asynchronous events. Fine low-level events are thus difficult to model and measure. Second, most physical side-channel cryptanalysis approaches require the leakage signal to be acquired at rates well beyond the device's clock rate; for multi-GHz CPUs, the requisite equipment is expensive, and the signals are difficult to probe. Finally, attack scenarios differ: the aforementioned small devices are often deployed into potentially-malicious hands, where they could be subjected to lengthy or invasive attacks; but for PCs, the typical scenario (short of theft) is where a physical attacker gains physical proximity for a restricted amount of time, and must operate surreptitiously.

Recently, a key extraction attack on PCs was demonstrated using the acoustic side channel, addressing all three barriers: using a chosen-ciphertext attack, the sound emanations of interest are made very robust, brought down to very low frequencies (tens or hundreds of kHz), and extended to long durations (hundreds of milliseconds), making it possible to record the leakage surreptitiously and non-invasively, by a cellphone microphone or from many meters away [GST14].

We thus study the question: *what other physical, non-invasive, cryptanalytic side-channel attacks can be effectively conducted on full-blown PC computers?*

1.2 Our results

We demonstrate key extraction of 4096-bit RSA and 3072-bit ElGamal keys from laptop computers of various models. The attacked software implementation is GnuPG [Gpg], a popular open source implementation of the OpenPGP standard [CDF+07]. The attacks exploit several side channels, enumerated below (along with examples of attack scenarios):

1. **Chassis potential.** We identify a new side channel: fluctuations of the electric potential on the chassis of laptop computers, in reference to the mains earth ground. This potential can be measured by a simple wire, non-invasively touching a conductive part of the laptop, and connected to a suitable amplifier and digitizer. (This, essentially, creates a ground loop through the laptop and measures its voltage.) The chassis potential, thus measured, is affected by ongoing computation, and our attacks exploit this for extracting RSA and ElGamal keys, within a few seconds.

   Scenarios: The probe wire can be fixed in advance in a location where the target laptop will be placed (e.g., a cluttered desk), or put in contact with the laptop by a passerby.

2. **Far end of cable.** The chassis potential can also be observed from afar, through any cable with a conductive shield that is attached to an I/O port on the laptop. For example, we demonstrated key recovery through a 10-meter long Ethernet cable, by tapping the cable shield at the remote Ethernet switch (see Figure 12). Similar observations apply to USB, VGA, HDMI, etc. Since only the shield potential is used, the attack is oblivious to the data passing through the cable, and works even if the port is disabled.

   Scenarios: While many users are careful about connecting suspicious devices (such as USB sticks) to the physical ports of their machines, they will routinely connect VGA display cables to their laptops.

Likewise, users often connect Ethernet cables to their laptop computers when an adequate firewall is configured or when the environment appears trustworthy. However, a simple voltage measurement device, perhaps located in the cabinet or server room to which the cable leads, could be capturing the leakage. This is hard to check, since Ethernet wiring and projectors' VGA cables are often hidden out of sight and cannot easily be tracked by the user.

3. **Human touch.** Surprisingly, the requisite signal can be measured, with sufficient fidelity, even through a human body. An attacker merely needs to touch the target computer with his bare hand, while his body potential is measured.

   Scenarios: The attacker positions himself in physical proximity to the target laptop and, under some ruse, touches it with his bare hand or a conducting pen (see Figure 13). Surreptitiously, the attacker carries the requisite equipment for measuring his body potential relative to some nearby grounded object. In the non-adaptive attack (see below), a few seconds' contact will suffice; in the adaptive attack, 1 key bit can be extracted approximately every 4 seconds of contact.

Note that the above attacks all rely on fluctuations in the PC device's ground (relative to the mains earth ground). This makes electrical mitigation difficult: the usual method for preventing voltage fluctuations, using bypass capacitors to shunt stray AC currents into the device's ground, does not apply to the device ground itself.

We also revisit two traditional physical side channels, and demonstrate their applicability to software running on PCs:

4. **Electromagnetic (EM).** We performed key extraction by measuring the induced EM emanations, using a near-field probe placed near the laptop.

   Scenarios: Electromagnetic probes are easily hidden in nearby objects. A glove, containing a concealed probe loop and hovering over the target laptop, would unveil its key within seconds.

5. **Power.** Likewise, we extracted keys by measuring the current draw on the laptop's power supply. Our attack works even though PCs use complex switching power supplies, which partially decouple the power source from the CPU load,[1], and moreover employ large capacitors, chokes, and shields for electromagnetic compatibility (EMC) compliance — all of which attenuate and disrupt the signals sought in traditional power analysis.

   Scenarios: A public charging station can be easily augmented with a current meter, logger, and transmitter. Even a regular power supply "brick" can be similarly augmented, and such laptop power supplies are often shared, offered to guests, or left unattended.

Our attacks require very low bandwidth, well below the laptop CPU's GHz-scale clock rate. We use two cryptanalytic approaches, based on known techniques and adapted to the target software:

Fast, non-adaptive MF attack. For both RSA and ElGamal key extraction, we can exploit signals circa 2 MHz (Medium Frequency band), using the "$n-1$" non-adaptive chosen-ciphertext simple-power-analysis attack of Yen et al. [YLMH05]. Key extraction then requires a few seconds of measurements.

Slow, adaptive VLF/LF attack. For RSA key extraction, we can exploit signals of about 15–40 kHz (Very Low Frequency / Low Frequency bands), using the adaptive chosen-ciphertext attack of [GST14]. Full 4096-bit RSA key extraction then takes approximately one hour, but is very robust to low signal-to-noise ratio.

Our results require careful choice and tuning of the signal acquisition equipment, to attain usable signal-to-noise ratio in the presence of abundant noise, delicate grounding, and impedance-matching considerations. We report these choices in detail and describe the requisite signal processing. We also analyze the code of GnuPG's mathematical library, showing why the specific chosen ciphertext creates exploitable, key-dependent leakage in this implementation.[2]

[1] In the realm of small devices, such similar decoupling has been proposed as an intentional countermeasure against power analysis [TB10].

[2] The combinations of side channel, attack technique, target algorithm, and target computer are too numerous to exhaustively demonstrate and discuss, especially due to the requisite analog and algorithmic tuning. This paper summarizes dozens

1.3 Vulnerable software and hardware

Hardware. We have tested various laptop computers, of different models, by various vendors. The signal quality varied dramatically, as did the relative quality between channels, the carrier frequencies of interest, the best placement of the probes or human hand, and the optimal grounding connection. Thus, manual calibration and experimentation were required. Generally, instruction-dependent leakage occurs on most laptops; key extraction is possible on many laptops, but the requisite signal-to-noise is not always present. Heavily-used laptops appear to exhibit stronger information-bearing signals.

GnuPG. For this case study, we focused on GnuPG version 1.4.15, running on Windows XP and compiled with the MinGW GCC version 4.6.2. This version of GnuPG was the most recent when our research was publicly announced. Following the practice of responsible disclosure, we worked with the authors of GnuPG to suggest several countermeasures and verify their effectiveness against our attacks (including those in [GST14]; see discussion therein as well as CVE-2013-4576 [MIT13]). GnuPG 1.4.16, released concurrently with the announcement of our results, contains these countermeasures.

Chosen ciphertext injection. Our key extraction attacks require chosen ciphertexts (either adaptive or non-adaptive, depending on the attack). As observed in [GST14], one way to remotely inject such ciphertexts into GnuPG is to send them as a PGP/MIME-encoded e-mail [ETLR01], to be automatically decrypted by the Enigmail [Eni] plugin for the Mozilla Thunderbird e-mail client. In the case of the non-adaptive attack, the attacker can provide the chosen ciphertext files to the victim (by any means or guise), and merely needs to conduct the measurement, for a few seconds, when those files are decrypted.

1.4 Related work

Simple and differential power analysis attacks were introduced by Kocher et al. [KJJ99], and applied to both symmetric and asymmetric ciphers, implemented on hardware such as smartcards, microcontrollers and FPGAs (see [And08, MOP07, KJJR11] and the references therein).

Clark et al. [CMR+13] observed that is it possible to use power analysis to identify, with high probability, the web pages loaded by a web browser on the target machine, by tapping the AC electrical outlet to which the target is connected.

Oren and Shamir [OS06] observed that the power line voltage on USB ports exhibits a distinct signature when OpenSSL RSA decryption executes, even when the port is disabled; this property is shared by our "far end of cable" channel. Schmidt et al. [SPK+10] observed leakage through voltage variations on the I/O pins of embedded devices.

Basic multiplication instructions were shown to have operand-specific leakage, in simulation [WS05] and embedded devices [CFR10] (though this was not demonstrated or exploited on PCs).

The electromagnetic side channel has been studied and exploited for smart-cards and FPGA's (e.g., [QS01, GMO01, AARR02]), including for RSA. More recently, Zajic and Prvulovic [ZP14] observed electromagnetic leakage from laptop and desktop computers (but did not show cryptographic applications). Timing attacks have been shown on software implementations of DSS, RSA, and ECDSA [Koc96, WT01, BB05, BT11]. Cache attacks [Ber05, Per05, OST06] were applied to GnuPG's RSA implementation [YF13].

1.5 Paper outline

The remainder of this paper is organized as follows. Section 2 establishes the presence of computation-dependent side-channel leakage via fluctuations of computers' chassis potential. Section 3 presents and analyzes the fast, non-adaptive MF attack. Section 4 recalls the slow, adaptive VLF/LF attack. Section 5 reports experimental key-extraction results, using both attacks, applied to the five aforementioned channels. Section 6 concludes and discusses countermeasures.

of successful key extraction configurations.

2 Computation-dependent chassis-potential leakage

The electric potential on a laptop computer's chassis (metal panels, shields and ports) is ideally equal to that of the mains earth ground potential, but in reality it fluctuates greatly. Even when the laptop is grounded (via its power supply connector or via shielded cables such as Ethernet, USB, VGA, DisplayPort or audio), there is non-negligible impedance between the grounding point(s) and other points in the chassis. Voltage, often 10mV RMS or more,[3] develops across this impedance, in part due to currents and electromagnetic fields inside the computer. Since the latter depend on the ongoing computation, a natural question is whether information about the ongoing computation can be learned by measuring the chassis potential.

In this section we observe elementary correlations between the chassis potential (observed in various ways) and the operations performed by the target machine. These operations will include both individual CPU operations as well as various public key operations such as decryption and signing. We show that it is possible for an attacker to obtain information about the operations executed by the target machine, as well as some information about the secret keys. Later (Section 5), we discuss the hardware setup in detail, describe additional leakage channels, and show that each of these channels can be used for key-extraction attacks on GnuPG's implementation of RSA and ElGamal.

2.1 Code-dependent leakage

We first demonstrate leakage of information about which instructions are executed by the CPU of the target computer. As in [GST14], we let the target computer run a simple program that executes (partially unrolled) loops containing one of the following x86 instructions: HLT (CPU sleep), MUL (integer multiplication), FMUL (floating-point multiplication), memory access (forcing L1 and L2 cache misses, so they reach DRAM), and REP NOP (short-term idle). Figure 1 shows a recording of one target computer's chassis potential, while this program is running; different types of operations can be easily distinguished. Leakage on other machine models is demonstrated in Figure 2. Moreover, similar effects can be observed by measuring EM emanations. Finally, similar effects can also be observed by touching the laptop's chassis and measuring the attacker's body potential (see Figure 3).

2.2 GnuPG key distinguishability

The leakage via chassis potential and electromagnetic emanations is also applicable to cryptographic code. In this section, we show a very simple (yet already troubling) form of leaked information: determining which of several randomly-generated secret keys was used by the target machine, for a signing or decryption operation. For brevity, in the remainder of this section we discuss chassis potential measurements; similar effects are also present when using the EM side channel. For ease of comparison, our evaluation follows the methodology of [GST14].

Distinguishing RSA secret keys. Figure 4 depicts the spectrogram of four GnuPG RSA signing operations, using different 4096-bit random keys (generated beforehand), on a fixed message. Each signing operation is preceded by a short CPU sleep (for visual separation).

The different signing keys can be clearly distinguished by their subtly different spectral signatures (which may itself be of practical interest). Another telling detail is visible: halfway through each signing operation, a transition appears at several frequency bands (marked with yellow arrows). This transition corresponds to the transition between exponentiation modulo the secret p to exponentiation modulo

[3] After filtering out the strong, but cryptanalytically useless, components at 50 Hz or 60 Hz.

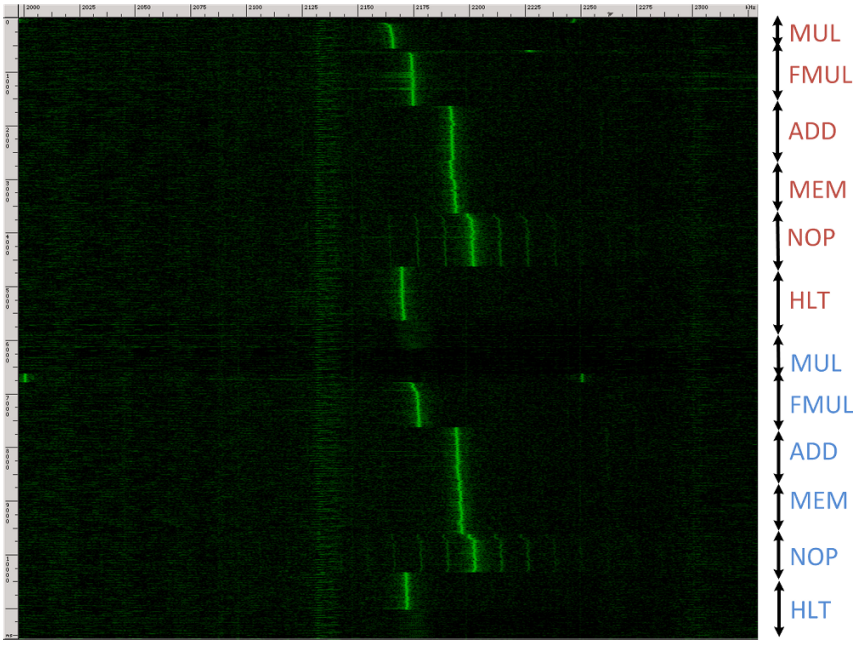
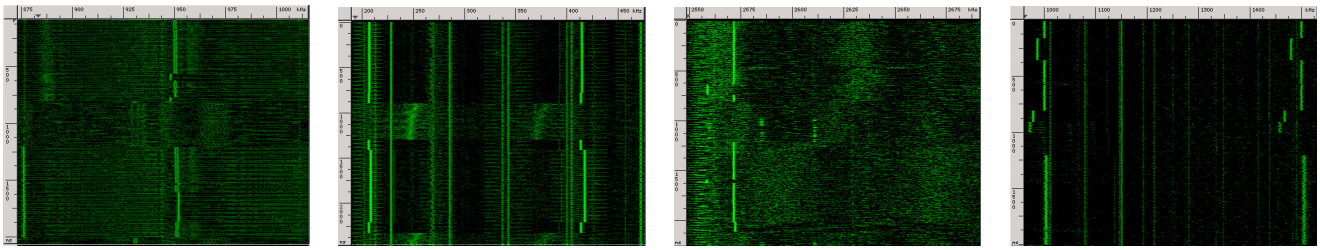


Figure 1: Frequency spectrogram of the chassis potential, while running different CPU operations using a Lenovo 3000 N200 laptop. The horizontal axis is frequency (2–2.3 MHz), the vertical axis is time (10 sec), and intensity is proportional to the instantaneous energy in that frequency band.

the secret q, in the RSA decryption implementation of GnuPG, which is based on the Chinese Remainder Theorem. We can thus spectrally observe internal, secret-dependent information within the signing operation.

ElGamal key distinguishability. Another popular public key encryption scheme is ElGamal encryption [ElG85]. Recall that in ElGamal encryption, the key generation algorithm consists of generating a large prime p, a generator α of $\mathbb{Z}_p^*$, and a secret exponent $a \in \mathbb{Z}_p^*$. The public key is $\mathsf{pk} = (p, \alpha, \alpha^a)$ and the secret key is $\mathsf{sk} = a$. Encryption of a message m outputs a pair (γ, δ), where $\gamma = \alpha^k \bmod p$ and $\delta = m \cdot (\alpha^a)^k \bmod p$ for random k. Decryption of (γ, δ) outputs $\gamma^{-a} \cdot \delta \bmod p$. Figure 5 presents a recording of four ElGamal decryptions, using a fixed message m and randomly-generated keys with 3072-bit primes p. As in the RSA case, the four decryptions are clearly visible, and the four different keys can be easily distinguished.



(a) Dell Latitude E6400 (b) Gateway W340UA (c) Lenovo ThinkPad T61 (d) HP HDX 18

Figure 2: Chassis measurements of various target computers performing MUL, HLT and MEM in this order. Note that the three operations can be distinguished on all machines.

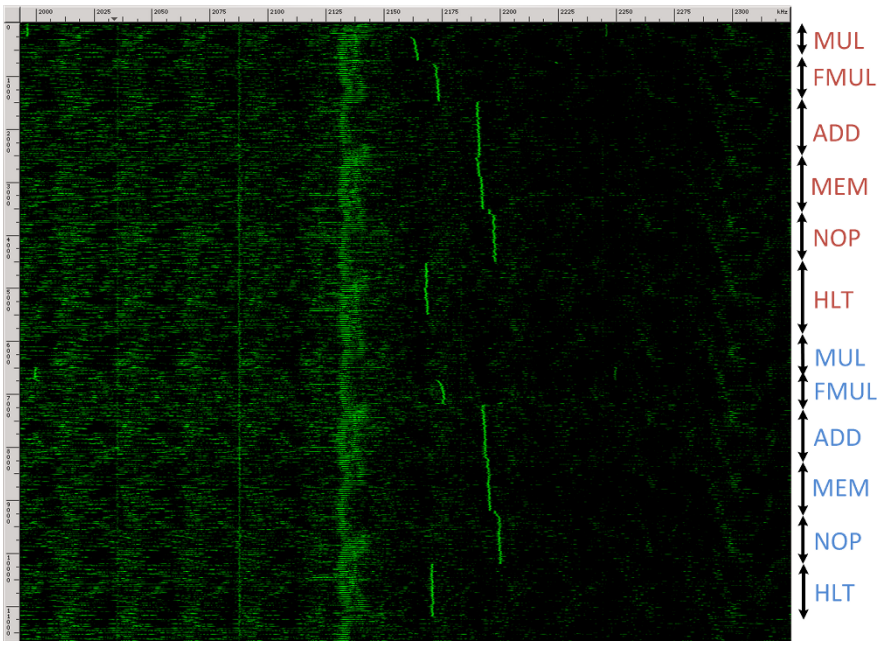


Figure 3: Frequency spectrogram (1.1 sec, 2-2.3 MHz) of a recording of different CPU operations obtained while measuring the attacker's body potential while touching an exposed part of the chassis of a Lenovo 3000 N200 laptop.

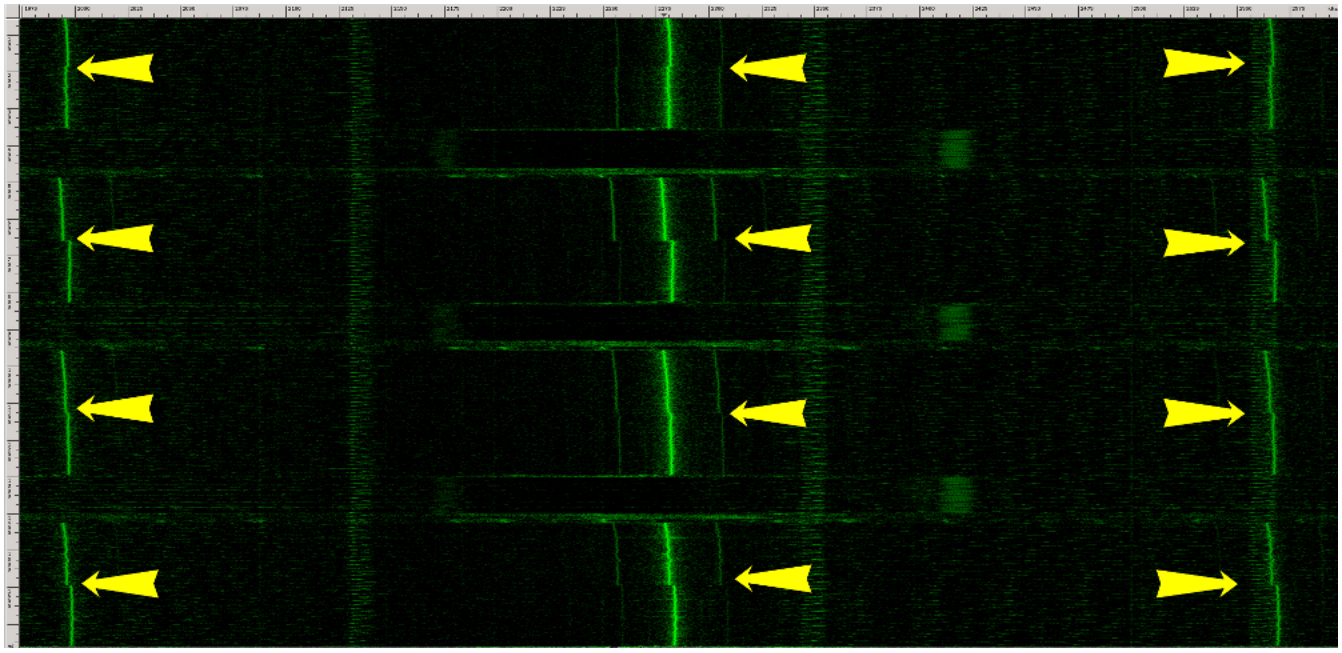


Figure 4: Chassis measurement (1.7 sec, 1.9–2.6 MHz) of four GnuPG RSA signatures executed on a Lenovo 3000 N200. The transitions between p and q are marked with yellow arrows.

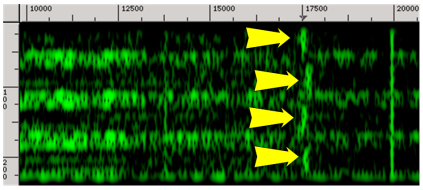


Figure 5: Chassis measurement (0.20 sec 0–20 kHz) of four ElGamal decryptions using the same message and randomly generated 3072 bit keys executed on a Lenovo ThinkPad T61.

3 Non-adaptive attack

We proceed to describe our cryptanalytic attack techniques (whose applicability will be shown in Section 5). The first technique is a non-adaptive chosen ciphertext attack using very few traces, following the simple power analysis of RSA (see [KJJ99, MDS99, Nov02, FV03, YLMH05, HMA+08], the surveys [MOP07, And08, KJJR11], and the references therein). We begin by reviewing the high-level modular exponentiation algorithm in GnuPG (Section 3.1), describe our attack (Section 3.2) exploiting this algorithm, and then analyze its success by recalling the inner squaring routines used by GnuPG (Section 3.3) and their behavior under the attack (Section 3.4).

3.1 GnuPG's modular exponentiation routine

To perform arithmetic on the large integers occurring in RSA and ElGamal, GnuPG uses an internal mathematical library called MPI (based on GMP [Gmp]). MPI stores large integers as arrays of *limbs*, i.e., 32-bit words (on the x86 architecture used in our tests).

We now review the modular exponentiation routine of GnuPG (as introduced in GnuPG v1.4.14), which is used for both RSA and ElGamal. GnuPG uses a variant of the square-and-multiply modular exponentiation algorithm, processing the bits of the exponent from the most significant bit to the least significant one. To mitigate a cache side-channel attack [YF13], GnuPG now always performs the multiplication operation in every loop iteration regardless of the exponent's bits (but only uses the result as needed). The pseudocode is given in Algorithm 1. The operation SIZE_IN_LIMBS(x) returns the number of limbs in the t-bit number x, namely $\lceil t/32 \rceil$. This top-level exponentiation routine suffices for the high-level description of our attack. For details about GnuPG's underlying squaring routines, necessary for understanding the attack's success, see Sections 3.3 and 3.4.

3.2 The attack algorithm

Since GnuPG 1.4.15 attempts to avoid correlation between the bits of the exponent and *high-level* operations (such as multiplication), we use a chosen ciphertext attack in order to create correlations between these bits and intermediate values computed inside the *low-level* operations inside GnuPG's modular exponentiation routine. Moreover, the chosen ciphertext will have an amplification effect, whereby numerous recursive calls will be similarly affected, resulting in a distinct leakage signal over a long time period. This is the key to conducting a MHz-scale attack on a GHz-scale computation.

The technique of Yen et al. [YLMH05] is particularly suitable. In the case of both RSA-CRT and ElGamal, it chooses a ciphertext c such that $a \equiv -1 \pmod{p}$ during the execution MODULAR_EXPONENTIATION (Algorithm 1) above. Within GnuPG, this ciphertext creates a correlation between the bits of the secret exponent b and the number of zero limbs of m, as described next. As we will analyze in Sections 3.3 and 3.4, the number of zero limbs inside m drastically affects the control flow inside the GnuPG basic squaring routine,thus creating discernible differences in the physical leakage.

Note that, for $a \equiv -1 \pmod{p}$, the value of m during the execution of MODULAR_EXPONENTIATION is always either 1 or -1 modulo p. Thus, the value of m in line 7 does not depend on the bits of b, and is always 1 modulo p (since $-1^2 \equiv 1^2 \equiv 1 \pmod{p}$). Consequently, the following correlation holds between the value of m at the start of the i-th iteration of the loop in line 5 and b_{i-1}.

- $\boldsymbol{b_{i-1} = 0}$**.** In this case the branch on line 8 was not taken; thus, the value of m at the start of the i-th iteration of the loop in line 5 is also $m = 1 \bmod p = 1$. Next, since GnuPG's internal representation does not truncate leading zeros, it holds that the value m sent to the squaring routine during the i-th iteration contains many zero limbs.
- $\boldsymbol{b_{i-1} = 1}$**.** In this case the branch on line 8 was taken, so the value of m at the start of the i-th iteration of the loop in line 5 is $m = -1 \bmod p = p - 1$. Since p is a random large prime, the value m sent to the squaring routine during the i-th iteration contains very few zero limbs.

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

Input: Three integers a, b and p in binary representation such that $b = b_1 \cdots b_n$.
Output: $m \equiv a^b \pmod{p}$.

```
1:  procedure MODULAR_EXPONENTIATION(a, b, p)
2:      if SIZE_IN_LIMBS(a) > SIZE_IN_LIMBS(p) then
3:          a ← a mod p
4:      m ← 1
5:      for i ← 1 to n do
6:          m ← m^2 mod p                    ▷ Karatsuba or grade-school squaring
7:          t ← m · a mod p                  ▷ Karatsuba or grade-school multiplication
8:          if b_i = 1 then
9:              m ← t
10:     return m
11: end procedure
```

We now proceed to describe the specific ciphertext choices required for our attack for both the RSA and ElGamal case.

Ciphertext choice for RSA-CRT. Recall that in the case of RSA decryption, GnuPG first computes $c^{d \bmod (p-1)} \bmod p$ and $c^{d \bmod (q-1)} \bmod q$, and then combines these via the Chinese Remainder Theorem. By choosing $c = n - 1$ where $n = pq$ is the public RSA modulus, the modular reduction in line 3 is always triggered, causing the value of a in line 5 to be $p - 1$ (i.e., -1 modulo p as desired).

Ciphertext choice for ElGamal. For ElGamal encryption, the prime modulus p is part of the public key, so we directly choose the ciphertext to be $p - 1$.

3.3 GnuPG's squaring routine

GnuPG's large-integer squaring routine combines two squaring algorithms: a basic quadratic-complexity squaring routine, and a variant based on a recursive Karatsuba multiplication algorithm [KO62]. The chosen combination of algorithms is based on the size of the operands, measured in whole limbs. We will first discuss the basic squaring algorithm and its key-dependent behavior, and then show how this behavior is preserved by the Karatsuba squaring.

GnuPG's basic squaring routine. The core side-channel weakness we exploit in GnuPG's code lies inside the basic squaring routine. The basic squaring routine used by GnuPG is a simple, quadratic-complexity "grade school" squaring, with optimizations for multiplication by limbs equal to 0 or 1, depicted in Algorithm 2.

Note how SQR_BASECASE handles zero limbs of a. In particular, when a zero limb of a is encountered, none of the operations MUL_BY_SINGLE_LIMB, ADD_WITH_OFFSET and MUL_AND_ADD_WITH_OFFSET are performed and the loop in line 9 continues to the next limb of a. Our attack exploits this, by causing the number of such zero limbs to depend on the current bit of the secret exponent, thus affecting the control flow in lines 3 and 11, and thereby the side-channel emanations.

GnuPG's Karatsuba squaring routine. The basic squaring routine described above is invoked via two code paths: directly by the modular exponentiation routine (Section 3.1) when the operand is small, and also as the base-case by the Karatsuba squaring routine. The latter is a variant of the Karatsuba multiplication algorithm [KO62], relying on the following identity:

$$a^2 = \begin{cases} (2^{2n} + 2^n)a_{\mathsf{H}}^2 - 2^n(a_{\mathsf{H}} - a_{\mathsf{L}})^2 + (2^n + 1)a_{\mathsf{L}}^2 & \text{if } a_{\mathsf{H}} > a_{\mathsf{L}} \\ (2^{2n} + 2^n)a_{\mathsf{H}}^2 - 2^n(a_{\mathsf{L}} - a_{\mathsf{H}})^2 + (2^n + 1)a_{\mathsf{L}}^2 & \text{otherwise} \end{cases}, \tag{1}$$

where $a_{\mathsf{H}}, a_{\mathsf{L}}$ are the most and least significant halves of a, respectively.

Algorithm 2 GnuPG's basic squaring code (see function sqr_n_basecase in mpi/mpih-mul.c).

Input: A number $a = a_k \cdots a_1$ of size k.
Output: a^2.

```
1: procedure SQR_BASECASE(a)
2:     if a_1 ≤ 1 then
3:         if a_1 = 1 then
4:             p ← a
5:         else                                                    ▷ a_i = 0
6:             p ← 0
7:     else
8:         p ← MUL_BY_SINGLE_LIMB(a, a_1)                       ▷ p ← a · a_1
9:     for i ← 2 to n do
10:        if a_i ≤ 1 then
11:            if a_i = 1 then                          ▷ (and if a_i = 0 do nothing)
12:                p ← ADD_WITH_OFFSET(p, a, i)                 ▷ p ← p + a · 2^(32·i)
13:        else
14:            p ← MUL_AND_ADD_WITH_OFFSET(p, a, a_i, i)    ▷ p ← p + a · a_i · 2^(32·i)
15:    return p
16: end procedure
```

3.4 Attack analysis

In this section we analyze the effects of our attack on SQR_BASECASE (Algorithm 2). Recall that in Section 3.2 we created a correlation between the i-th bit of the secret exponent b_i and the number of zero limbs in the operand m of the squaring routine during iteration $i+1$ of the main loop of the modular exponentiation routine. Concretely, for the case where $b_i = 1$, we have that $m = -1 \bmod p = p - 1$ is a random-looking number containing several thousand bits (2048 bits for the case of RSA and 3072 bits for the case of ElGamal). Conversely, for the case where $b_i = 0$, we have that $m = 1 \bmod p = 1$ and, since GnuPG does not truncate leading zeros, the representation of m is a large number (2048 bits for the case of RSA and 3072 bits for ElGamal), all of whose limbs are 0 (except for the least significant).

The code of GnuPG passes m directly to the Karatsuba squaring routine. Notice that for the case where $b_i = 1$, since m is a random-looking number, this property of m will be preserved in all 3 recursive calls (computing the three squaring operations in Equation 1). Similarly, for the case of $b_i = 0$, we have that $m = 1$, meaning $m_{\mathsf{H}} = 0$ and $m_{\mathsf{L}} = 1$. Thus, the second case of Equation 1 will always be taken, again preserving the structure of m as having mostly zero limbs, in all 3 recursive calls.

When reaching the recursion's base case, we have the following dependence on b_i. If $b_i = 0$, then the values of the operand of SQR_BASECASE during iteration $i+1$ of the main loop of MODULAR_EXPONENTIATION (in all branches of the recursion) will have almost all of their limbs equal to zero. Conversely, if $b_i = 1$, then the values of the operand of SQR_BASECASE during iteration $i+1$ of the main loop of MODULAR_EXPONENTIATION in all branches of the recursion will be random-looking. Thus, by (indirectly) measuring the instructions executed by SQR_BASECASE, we shall be able to deduce the number of zero limbs in its operand and thus obtain b_i.

Confirming this, Figure 6 presents the number of zero limbs in the operand of SQR_BASECASE during each iteration of the main loop of the modular exponentiation routine using our chosen ciphertext and five randomly generated keys. The values of the first 50 bits of the exponent are clearly visible (many zero limbs during the ith iteration implies that the $i-1$th bit of the secret exponent is 0).

Next, recall the effect of zero limbs in the operand on the code of SQR_BASECASE. Note that the control flow in SQR_BASECASE depends on the number of non-zero limbs in its operand. The drastic change in the

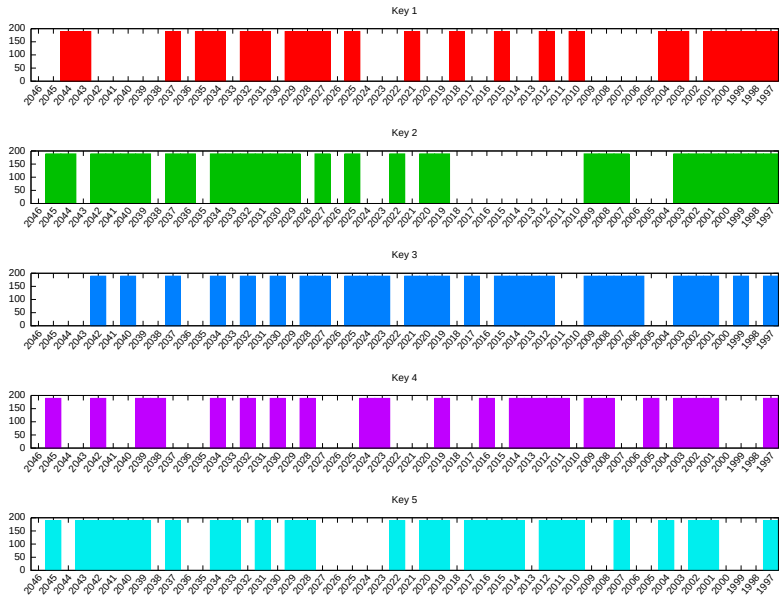


Figure 6: Number of zero limbs in the operand of SQR_BASECASE(a) during an execution of the modular exponentiation routine, using our attack and randomly generated keys. Whenever the $i-1$th bit of the secret exponent is 0, there are 189 zero limbs during the squaring; otherwise there are none.

number of zero limbs in the operand of SQR_BASECASE is detectable by our side-channel measurements. Thus, we are able to leak the bits of the secret exponent by creating the correlation between its bits and the number of zero limbs in the operand of SQR_BASECASE, using our carefully chosen cipher text.

4 Adaptive attack

Our other cryptanalytic technique is an adaptive chosen-ciphertext side-channel attack, which extracts, bit by bit, the prime modulus used during the CRT-based RSA modular exponentiation routine. Our attack on GnuPG was introduced in [GST14] (following [BB05]) for the GnuPG acoustic side channel. For self-containment, we give an overview of the attack here. (This attack is not applicable to ElGamal, since the prime modulus p is public.)

The attack recovers the bits of $p = p_1 \cdots p_k$ iteratively, starting with the MSB p_1. Once we learn all of p, we know the factorization of n. Moreover, after recovering just the top half of the bits of p, it is possible to use Coppersmith's attack [Cop97] to recover the remaining bits.

Ciphertext choice for RSA. In GnuPG, the MSB is always set, i.e., $p_k = 1$. Assume that we have already recovered the topmost $i-1$ bits of p. To extract the next bit p_i, we check the two hypotheses about its value, by requesting decryption of an adaptively chosen ciphertext $g^{i,0}+n$, where $g^{i,0} = p_1 \cdots p_{i-1}10\cdots 0$ (k bits in total). Let $n = pq$ be the public RSA modulus. Consider the RSA decryption of $g^{i,0} + n$. Two cases are possible, depending on p_i.

- $\boldsymbol{p_i = 1}$. Then $g^{i,0} < p$. The ciphertext $g^{i,0} + n$ is passed as the variable a to Algorithm 1. Since $n = pg$ is the product of two equal-sized primes, $g^{i,0} + n$ has a larger limb count than p. This triggers the modular reduction of a in line 3 of Algorithm 1, which returns $g^{i,0}$, resulting in a being a k-bit number having mostly zero limbs. Next, a is passed to the multiplication routine in line 7.
- $\boldsymbol{p_i = 0}$. Then $p < g^{i,0}$ and as in the previous case, $g^{i,0} + n$ is passed as the variable a to Algorithm 1 triggering the modular reduction of a in line 3. Since $g^{i,0}$ and p share the same topmost $i-1$ bits, we

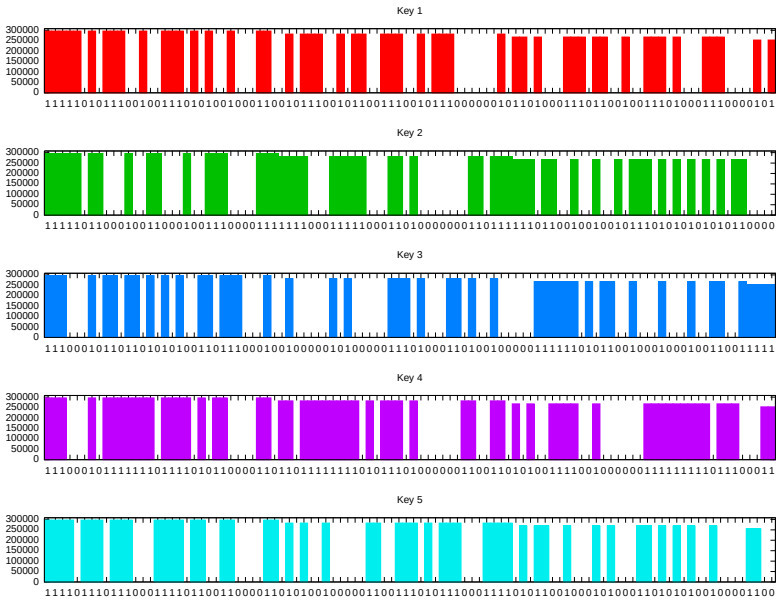


Figure 7: Number of zero limbs in the second operand of MUL_BASECASE(a, b) during an execution of the modular exponentiation routine with the secret prime p using our attack and randomly generated keys. The correct bit values are indicated on the horizontal axis of each key.

have that $g^{i,0} < 2p$, and the reduction results in $a = g^{i,0} - q$, which is a $(k - i)$-bit random-looking number. This is then passed to the multiplication routine in line 7.

Code analysis. We now present a high-level analysis of how our bit-by-bit chosen ciphertext attack affects the code of GnuPG. Using the above described ciphertexts, the second operand of the multiplication routine during the entire execution of the main loop of the modular exponentiation routine, will be either full-size and repetitive or shorter and random-looking (depending on the value of p_i). [4]

GnuPG's uses two algorithms for large integer multiplication: the Karatsuba-based multiplication algorithm and the grade-school (quadratic time) multiplication algorithm. GnuPG's variant of Karatsuba recursive multiplication relies on the identity $ab = (2^{2n} + 2^n)a_\mathsf{H}b_\mathsf{H} + 2^n(a_\mathsf{H} - a_\mathsf{L})(b_\mathsf{L} - b_\mathsf{H}) + (2^n + 1)a_\mathsf{L}b_\mathsf{L}$, where $a_\mathsf{H}, b_\mathsf{H}$ are the most significant halves of a and b, respectively, and $a_\mathsf{L}, b_\mathsf{L}$ are the least significant halves of a and b, respectively. We thus see that GnuPG's Karatsuba multiplication preserves the structure of a as being either random-looking or containing many zero limbs. That is, if a is random-looking, then $a_\mathsf{H}b_\mathsf{H}$, $(a_\mathsf{H} - a_\mathsf{L})(b_\mathsf{L} - b_\mathsf{H})$ and $a_\mathsf{L}b_\mathsf{L}$ are random-looking as well. Conversely, if a contains mostly zero limbs, the values of $a_\mathsf{H}b_\mathsf{H}$, $(a_\mathsf{H} - a_\mathsf{L})(b_\mathsf{L} - b_\mathsf{H})$ and $a_\mathsf{L}b_\mathsf{L}$ also contain mostly zero limbs.

Next, when the recursion reaches its base case (when multiplying numbers of 15 limbs or less), GnuPG passes a to the basic multiplication routine. GnuPG's basic multiplication routine is implemented similarly to Algorithm 2. In particular, it includes optimizations for zero limbs similar to lines 3 and 11 of Algorithm 2. Thus, we are able to leak the bits of p, one bit at a time, by creating a correlation between the current bit of p and the number of zero limbs in the second operand of GnuPG's basic multiplication routine using our chosen ciphertexts.

Confirming this, Figure 7 presents the number of zero limbs in the second operand of GnuPG's basic multiplication routine during an execution of the modular exponentiation routine using our carefully chosen cipher texts and five randomly generated 4096-bit keys. The values of the first 100 bits of p are clearly visible (many zero limbs implies that the value of the respective bit is 1).

[4] Recent GnuPG implementations use the side-channel mitigation technique of always multiplying the intermediate results

The above is a high-level description of the adaptive attack. In order to achieve full RSA key extraction, improvements are required to the basic attack algorithm. See [GST14] for details.

5 Empirical key-extraction results

In this section we discuss our empirical results for the various side channels. For each channel, we review the empirical setup and describe successful key extraction experiments.

5.1 Chassis-potential attack

5.1.1 Setup

As discussed in Section 2, there are computation-dependent fluctuations of the electric potential on the chassis of laptop computers. We measured the electric potential of the laptop's chassis by touching it with the simplest possible probe: a plain wire, 80 cm long. The wire is pressed against the chassis by hand, or (for longer attacks) using an alligator clip. The wire's potential is then measured through an amplifier, filters, and a digitizer (whose choice, discussed below, depends on the attack).

Grounding. The attack measures the voltage between the room's mains earth ground potential and the target computer's chassis potential. Put otherwise, we create a ground loop which includes the laptop chassis and the amplifier, and then measure the voltage across the amplifier's input impedance (and thus its complement: the voltage across the laptop chassis). Thus, correct grounding is essential to maximizing the signal-to-noise ratio. The measurement is done in single-ended mode, in reference to the mains earth ground potential, using low-impedance ground connection to the amplifier and digitizer. The target laptop is grounded through one of its shielded I/O ports (we used a VGA cable to a grounded screen, or a USB cable to a grounded printer). If the target laptop's grounding is removed, but the laptop is still connected to 3-prong (grounded) AC-DC power supply, the signal-to-noise ratio typically degrades.[5] With a 2-prong (ungrounded) power supply and no other ground connections, the signal-to-noise ratio is very low and our attacks do not work.

Chassis probe placement. The chassis of modern laptops, while made mostly of metal (for EMI shielding), is typically covered with non-conductive plastic. However, many IO ports, such as USB, Ethernet, VGA, DisplayPort and HDMI, typically have metal shielding which is connected to the chassis or PCB ground, and thus can be probed by the attacker. Also, metal heatsink fins are often easily reachable through the exhaust fan grill. Heuristically, the best results are achieved when the chassis is probed close to the CPU and its associated voltage regulator circuitry,[6] and if the laptop's ground connection is distant from the probing point.

Chassis potential or EM? To ascertain that we were indeed measuring chassis potential rather than stray electromagnetic fields, we broke the direct galvanic connection between the probe wire and the laptop's chassis, by inserting a sheet of paper in-between. As expected, this always resulted in severe signal attenuation.

5.1.2 Non-adaptive chassis-potential attack

MF measurement equipment. The non-adaptive attack exploits signals in the Medium Frequency (MF) frequency band, on the order of 2 MHz. To measure these signals, we connected the probe wire

by the input; but this *helps* our attack, since it doubles the number of multiplications.

[5] 3-prong laptop AC-DC power supplies typically do *not* have a low-resistance path between the grounding prong and the DC power plug.

[6] That is, similarly to the source of acoustic emanations [GST14].

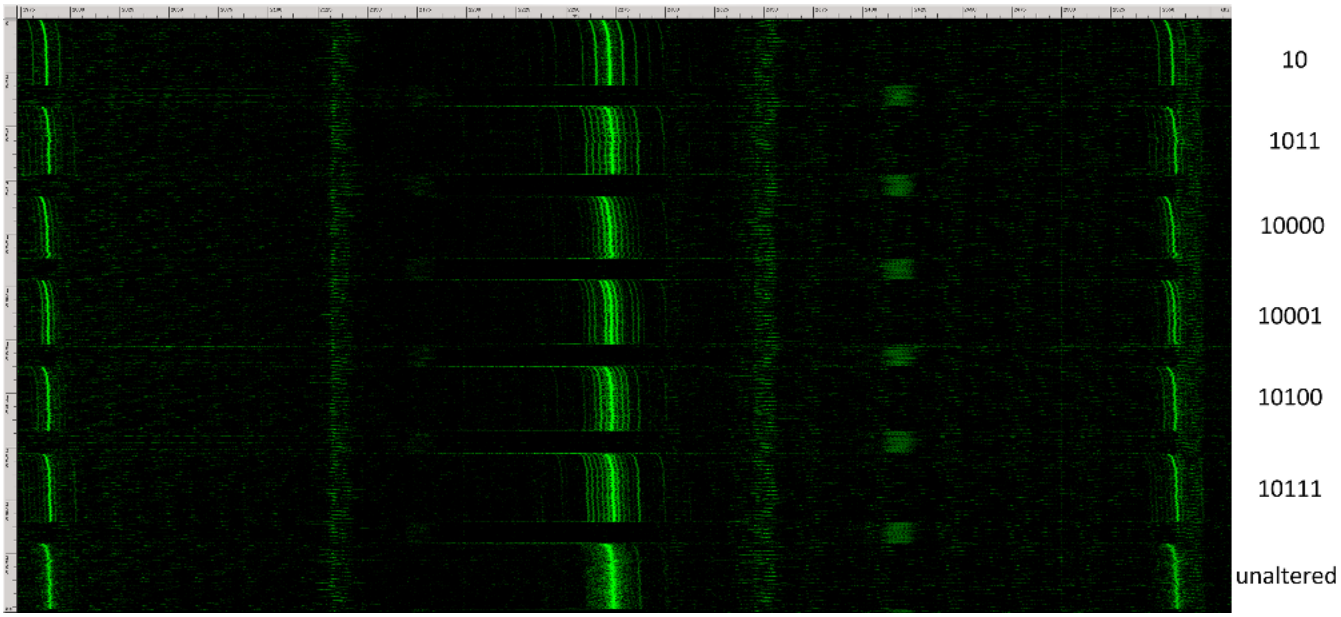


Figure 8: Chassis measurement (2.5 sec, 1.9–2.6 MHz) of seven GnuPG RSA decryptions executed on a Lenovo 3000 N200 laptop. In the first 6 cases, exponents (d_p and d_q) are both overridden to be the 2048-bit number obtained by repeating the pattern written to the right. In the last case, the exponent are unaltered. In all cases, the moduli p and q are the same and the ciphertext is set to $n - 1$.

to a 16 kHz high-pass filter, followed by a high-input-impedance low-noise amplifier (Femto HVA-200M-40-F, 40 dB gain). The amplified signal was then low-pass filtered at 5 MHz and digitized at 12 bits and 10 Msample/sec (National Instruments PCI 6115).

Exponent-dependent leakage. RSA and ElGamal decryptions both use modular exponentiation of the ciphertext to a secret exponent. We now examine how different exponents affect the chassis leakage. Figure 8 demonstrates seven RSA signing operations using different secret exponents. The different exponents can be easily distinguished. Similar results are observed for ElGamal decryption, though the exponentiation is shorter and thus its frequency spectrum can be characterized less accurately. **Analyzing the signal.** The signals presented in Figure 8 provide the first indication as to how the different exponents might be distinguished. For periodic exponents, the leakage signal spectrum takes the form of small, distinct side lobes centered around a dominant frequency peak. This indicates that the bits of the exponents manifest themselves as modulations on a central sinusoidal carrier wave. Further analysis reveals that the carrier is frequency-modulated, meaning that its instantaneous frequency changes slightly in accordance with the current bit of the exponent. Thus, in order to recover these bits, we obtain the dominant instantaneous frequency as a function of time, by applying FM demodulation digitally. The signal is first filtered using a 30 kHz band-pass filter centered at the carrier frequency. Next, the signal is demodulated using a Discrete Hilbert Transform. Additional filtering is then performed on the demodulated signal, to suppress high frequency noise and to compensate for a slow frequency drift of the carrier wave. Figure 9 shows an example of a fully demodulated leakage signal; the correlation with the secret key bits can clearly be seen.

Key extraction. Theoretically, it should be possible to extract the entire secret key from a single demodulated trace, provided the measurement is robust and has a high signal-to-noise ratio. However, we observed a periodic interrupt (marked in Figure 9), which disrupted the key extraction. The interrupt

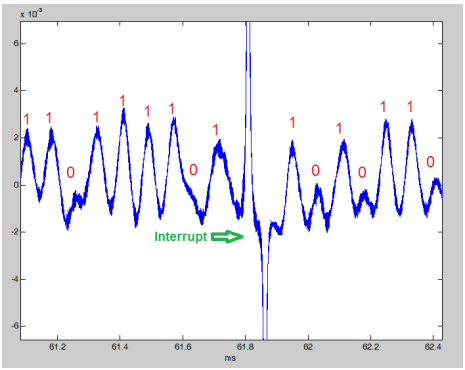


Figure 9: Frequency demodulation of a segment of the leakage signal obtained during a decryption operation using a randomly generated 4096-bit RSA key. Note the correlation between the signal and the secret key bits.

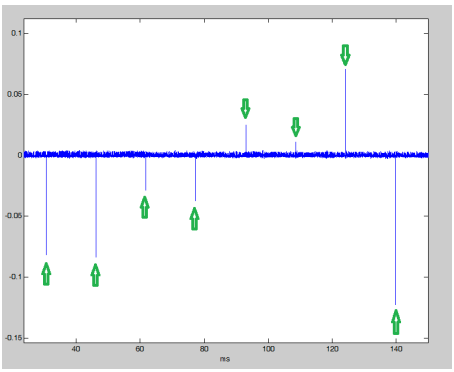


Figure 10: Frequency demodulation of the leakage signal during a decryption operation using a randomly generated 4096-bit RSA key. The interrupts, occurring every 15 milliseconds, are marked by green arrows.

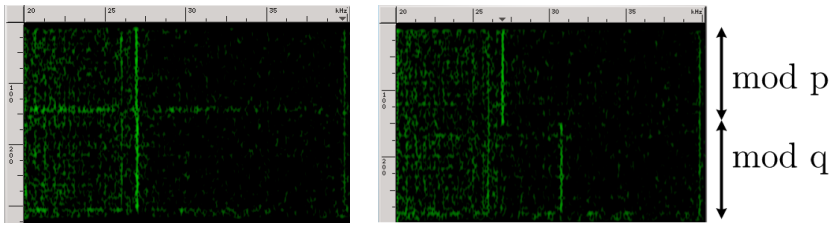
manifests as a large frequency fluctuation in the carrier every 15 milliseconds (Figure 10), corresponding to the 64 Hz timer interrupt frequency on the target laptop. These interrupts systematically occur at the same time during the decryption process (when decryption is deterministically synchronized to the OS scheduler), and thus disrupt similar bit offsets in repeated decryptions. Fortunately, the inherent jitter is sufficient so that, over a few measurements, every bit is (with high probability) undisturbed in some sample.

On the other hand, jitter creates a difficulty in aligning the multiple traces. We thus break each trace, post-modulation, into multiple time segments. The segments are then aligned via correlation, and averaged. This results in an interrupt-free aggregate trace, with very high SNR. The bits are then extracted using a peak detection algorithm.

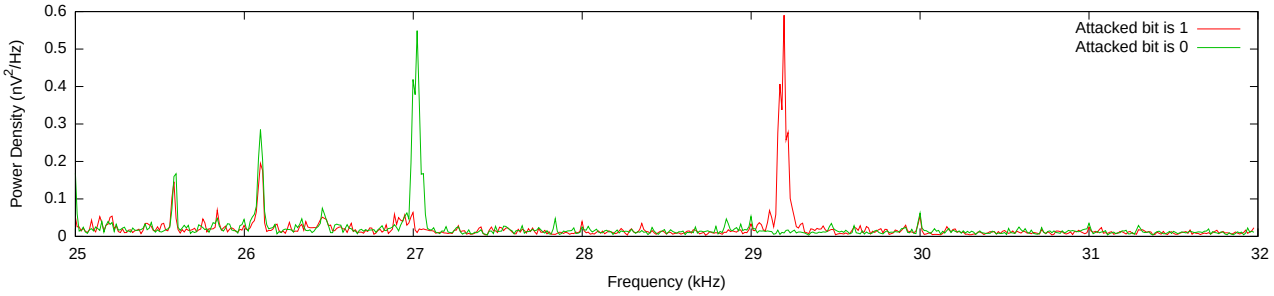
Non-adaptive RSA key extraction. Applying our non-adaptive attack on a randomly generated 4096-bit RSA key while measuring the chassis potential of a Lenovo 3000 N200 laptop during 6 decryption operations, each lasting 0.35 sec, we have directly extracted 2044 out of 2048 bits of d_p thereby extracting the key.[7] The laptop was powered by a 3-prong AC-DC power supply (Lenovo, 90W, model 42T4428), without additional connections.

Non-adaptive ElGamal key extraction. Attacking the exponentiation in ElGamal decryption, and

[7] The first few bits of p are harder to measure, due to stabilization time.



(a) attacked bit is $q_i = 0$ (b) attacked bit is $q_i = 1$



(c) Frequency spectra of the second modular exponentiation

Figure 11: Chassis measurement of RSA decryption for various values of the attacked bit executed on a Lenovo ThinkPad T60.

applying similar cryptanalytic and signal analysis, we extracted all but 2 of the bits of the secret exponent from a randomly generated 3072-bit ElGamal key by measuring the chassis potential of a Lenovo 3000 N200 laptop during 9 decryption operations, each lasting 0.1 sec

5.1.3 Adaptive chassis-potential attack

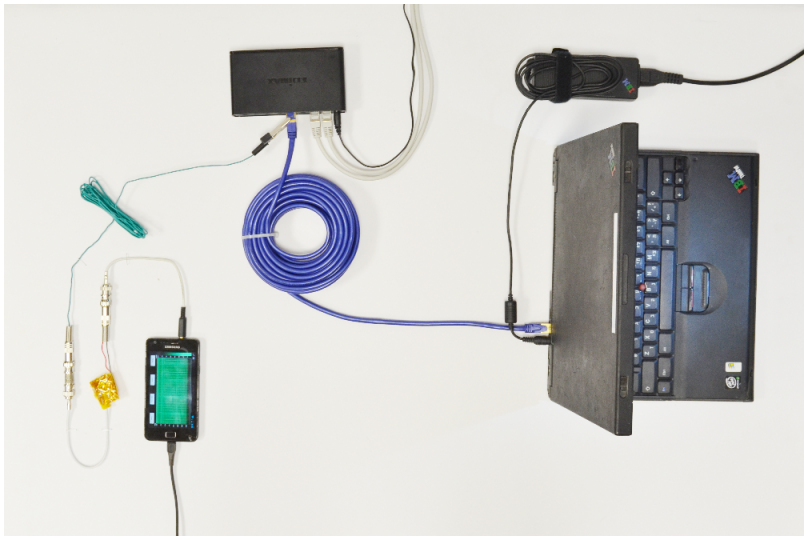
We now discuss results obtained using the adaptive attack described in Section 4. While this attack requires more decryption operations in order to recover the key, it utilizes lower frequency signals and requires much lower signal-to-noise ratio than the non-adaptive attack.

VLF/LF measurement equipment. The adaptive attack can exploit very low bandwidth signals, in the VLF and LF frequency bands: in the order of 15–40 kHz (depending on the laptop model). To measure these signals, we used a more compact, higher dynamic range, measurement chain. The probe was directly connected to a high-input-impedance low-noise amplifier (customized Brüel&Kjær 5935, usually set to 40 dB gain). The amplified signal was high-pass filtered at 10 kHz, and digitized at 16 bits and 200 Ksample/sec (National Instruments MyDAQ). Grounding and probe placement considerations are the same as for the non-adaptive attack.

Analyzing the leakage signal. Recall that GnuPG's RSA code performs modular exponentiation modulo p followed by a modular exponentiation modulo q. Figure 11(a) shows a typical recording of RSA decryption when the value of the attacked bit of q is 0, and Figure 11(b) shows a recording of RSA decryption when the value of the attacked bit of q is 1.[8] Several effects are shown in the figures. As in figure 4, the transition between p and q is clearly visible in Figures 11(a) and 11(b). Note, then, that the signatures of the modular exponentiation using the prime q (the second exponentiation) are quite different in Figures 11(a) vs. 11(b). This can be seen more clearly in Figure 11(c), which summarizes the aforementioned spectral signatures by taking the median, over time, for each. This clear difference is used to extract the bits of q, as explained in Section 4.

[8] Here we attack the exponentiation modulo q, to avoid the stabilization effects in the first exponentiation, modulo p.

(a) Measuring shield potential at the far side of an Ethernet cable (probed at the Ethernet switch).



(b) Adaptive "far end of cable" attack using a mobile phone. The chassis potential of the laptop is measured from the far side of an Ethernet cable (blue, 10 m long). An alligator clip connected to a plain wire (green) taps the shield of the Ethernet cable where it connects to an Ethernet switch. The signal passes through a passive filter (orange) and then, via a coax cable (white), into the microphone/earphone jack of the phone (Samsung Galaxy S II), where it is amplified and digitized. The phone itself is grounded to mains earth via its USB port. It is possible to perform the adaptive attack using this setup.

Figure 12: "Far end of cable" attacks.

Adaptive RSA key extraction. By connecting the VLF/LF measurement setup to a Lenovo ThinkPad T60, powered by the 3-prong AC-DC power supply without additional connections,[9] we directly extracted the 1024 most significant bits of the secret prime q of a randomly-generated 4096-bit RSA key in approximately 1 hour. By Coppersmith's technique, this results in full key extraction. (Alternately, it is possible to continue the attack and recover all the bits of q directly.)

5.2 "Far end of cable" attack

The electric potential on a laptop's chassis and external ports can measured from far away. When a cable is plugged into one of the laptop's many IO ports (such as USB, Ethernet, VGA, DisplayPort and HDMI), the port's shield typically contacts a plug shield, which in turn is connected to a conductive cable shield running the length of the cable. Thus, one can measure the chassis potential from the *far* side of cables connected to the aforementioned ports. Ethernet cables, for examples, often span long distances, across and between building floors. An attacker who gains access to the far side of the cable (see Figure 12(a)), or taps the shield along the way, can measure the approximate chassis potential.

Crucially, the attacker's only point of contact is the far side of the cable shield. The attack does not utilize the data transmitted over the cable, and is oblivious to whether the port is even enabled.

We conducted the attacks by connecting the target laptop, through a 10-meter long shielded CAT5e Ethernet cable, toa Gigabit Ethernet switch (EDIMAX ES-3308P). We attached a plain wire to the cable

[9] Grounding the laptop to mains earth, via some port, would improve the signal quality (see Section 5.1.1); but the adaptive attack is sufficiently robust to not require this.

(a) Non-adaptive "human touch" attack through a metal pen touching the heatsink fins. The wristband is connected to the probe wire.

(b) Adaptive "human touch" attack by bare hand. The wristband is connected to the probe wire.

Figure 13: "Human touch" attacks.

shield via a clip, on the switch side (see Figure 12(a)). Its potential was measured and analyzed as follows.

Non-adaptive RSA key extraction. Using the MF measurement setup (see Section 5.1.2), we measured the chassis potential of a Lenovo 3000 N200 laptop through the shield of the Ethernet cable. The laptop was powered by the 3-prong AC-DC power supply, and was also connected via a shielded VGA cable to a grounded monitor (Dell 2412M). We directly extracted 2042 out of 2048 bits of d_p, by observing the shield's potential during 5 decryption operations (each lasting 0.35 sec). Thus, by searching for the remaining 6 bits of d_p, the complete key is extracted.

Similar results were obtained by monitoring the far side of a (dangling) USB cable.

Non-adaptive ElGamal key extraction. Applying our non-adaptive attack to ElGamal decryption, we extracted all but 3 of the bits of the secret exponent, from a randomly-generated 3072-bit ElGamal key, by observing the shield's potential during 4 decryption operations (each lasting 0.1 sec).

Adaptive RSA key extraction. For this experiment, we attacked a Lenovo ThinkPad T60 laptop, through an Ethernet cable, whose shield potential was measured using the VLF/LF measurement equipment (see Section 5.1.3). The laptop was powered by the 3-prong AC-DC power supply, without additional connections. In this setting, we extracted the 1024 most significant bits of the secret prime q (and thus, by Coppersmith's technique, full key extraction) in 1 hour. Again, similar results were obtained by monitoring the far side of a (dangling) USB cable. Finally, on some machines, this attack only requires 15 kHz of analog bandwidth in order to extract the key making it possible to execute using almost any commodity sound recording equipment such as a mobile phone (see Figure 12(b)).

5.3 "Human touch" attack

On many laptops, the chassis potential can be sensed indirectly through a human body. An attacker can sense the chassis potential by merely *touching* a conductive part of the laptop chassis with his hand.[10] This affects the electric potential of the attacker's body (assuming suitable insulation, e.g., nonconductive floor or shoes). Surreptitiously, the attacker can measure the electric potential induced on his own body, using a concealed probe, in reference to some nearby conductive grounded surface in the room. Even this circuitous measurement, through an ill-characterized, high-impedance path, can suffice for key extraction.

Non-adaptive RSA key extraction. Using the MF measurement setup (see Section 5.1.2), we measured the chassis potential of a Lenovo 3000 N200 laptop indirectly, though the body of a volunteer (one of the authors). The volunteer held a paperclip or a metal pen in his hand, and briefly touched it to the

[10] The attack is especially effective in hot weather, since sweaty fingers offer lower electrical resistance.

laptop's heatsink fins, which are easily reachable through the exhaust vent.[11] Concurrently, the volunteer's body potential was measured using the MF measurement equipment (see Section 5.1.2), via a probe wire attached to a conductive wristband on his other wrist (see Figure 13(a)). The laptop was powered by the 3-prong AC-DC power supply, with no further connection.

Applying our non-adaptive attack and the signal analysis techniques from Section 5.1.2 on a randomly generated 4096-bit RSA key, we directly extracted 2042 out of 2048 bits of d_p by observing the volunteer's body potential (while holding the paperclip against the laptop's heatsink) during 6 decryption operations, each lasting 0.35 sec. Thus, by searching for the remaining 6 bits of d_p, the complete key is extracted.

Non-adaptive ElGamal key extraction. Applying our non-adaptive attack and the signal analysis techniques from Section 5.1.2 on a randomly generated 3072-bit ElGamal key, we were able to extract all but 3 of the bits of the secret exponent by observing the volunteer's body potential (while holding the paperclip against the laptop's heatsink) during 16 decryptions, each lasting 0.1 sec.

Adaptive RSA key extraction. The adaptive attack, being more robust (due to relying on lower-frequency, longer-duration signals), succeeded with unaided finger touch. A patient volunteer touched the chassis (specifically, VGA connector shield) of a Lenovo ThinkPad T61 with his fingers. The volunteer's body potential was measured using the VLF/LF measurement equipment (see Section 5.1.3), through the aforementioned wristband, and the laptop was connected as above. On some ThinkPad models, this attack can even be mounted by simply touching the rubber-coated LCD cover (see Figure 13(b)).

In this setting, we directly extracted the topmost 1024 bits of the prime q of a randomly generated 4096-bit RSA key from in one hour, thereby (via Coppersmith's technique) completely extracting the key.

5.4 Electromagnetic (EM) attack

Next, we studied EM emanations from laptop computers, in the MF band (approximately 2 MHz). We used a near-field magnetic probe (Rohde&Schwarz 7405901, 6 cm diameter, 50Ω impedance). The signal was low-pass filtered at 5 MHz and amplified using an impedance-matched low-noise amplifier (a customized Mini-Circuits ZPUL-30P). This is digitized at 12 bits and 10 Msample/sec (National Instruments PCI 6115).

EM probe placement. The placement of the EM probe relative to the laptop greatly influences the measured signal and noise. We wish to measure EM emanations close to the CPU's voltage regular, located on the laptop's motherboard, yet without mechanical intrusion. Luckily, most of the components covering the motherboard, such as the keyboard and its bezel cover, are almost transparent to EM signals. Concretely, hovering over the rear-left corner often yields the best signal.

Non-adaptive RSA key extraction. Applying our non-adaptive attack to a 4096-bit RSA key, we directly extracted 2046 out of 2048 bits of d_p, by measuring the EM emanations from a Lenovo 3000 N200 laptop during 5 decryption operations, each lasting about 0.35 sec.

Non-adaptive ElGamal key extraction. Using the same experimental setup and applying our non-adaptive attack to a randomly generated 3072-bit ElGamal key, we were able to extract all but 3 of the bits of the secret exponent by measuring the EM emanations from a Lenovo 3000 N200 laptop during 16 decryption operations, each lasting about 0.1 sec.

5.5 Power analysis attack

Finally, we revisited the classic power analysis channel, and analyzed the current draw fluctuations on the power supply of the target laptop in the VLF/LF frequency bands. Specifically, we placed a 0.5Ω resistor

[11] The heatsink fins provide a particularly strong signal, and the paperclip merely bypasses the mechanical obstruction of the plastic vent grill, a few millimeters deep. The attack is also possible by touching fully exposed metal connectors, such as I/O port shields, but in that case the signal is then weak and necessitates numerous measurements, so we applied the more robust adaptive attack (discussed next).

in series with the laptop's power supply, on the low ("ground") supply rail. For a typical laptop load of 2–5 A at 16–20 V, the voltage on the resistor was around 2 V.

Adaptive RSA key extraction. We measured the voltage on the resistor using a National Instruments MyDAQ device though a 150 kHz low-pass filter. We directly extracted the topmost 1024 bits of the prime q of a randomly generated 4096-bit RSA key, from a Lenovo ThinkPad T61, in one hour.

6 Conclusion

While physical side-channel attacks have proven very effective at breaking cryptosystems, most research attention has focused on small and relatively simple devices. This paper demonstrated that PC systems too are vulnerable, despite their complexity, noise, and challenging electrical characteristics. Moreover, PCs can be attacked by mere touch or from afar by almost any wired connection.

Following our observations, several software countermeasures were proposed and incorporated into GnuPG 1.4.16 and libgcrypt 1.6.:

Ciphertext normalization. Our adaptive attack requires a careful selection of ciphertexts that are slightly larger than p (but have the same limb count as p) to undergo reduction modulo p. However, for such ciphertexts, the modular exponentiation routine used by GnuPG will not normally reduce the ciphertext modulo p. We force this modular reduction to occur by padding the ciphertext with the public RSA moduli n or by padding additional zeros. Thus, one immediate countermeasure to our adaptive attack is, before decrypting a ciphertext c, to compute $c' = c \bmod n$ and then strip c of any leading zeros. This prevents the modular reduction in Line 3 of Algorithm 1, essential to the adaptive attack, from ever being executed.

Ciphertext randomization. A common countermeasure against chosen-ciphertext attacks is ciphertext randomization. For the case of RSA, such blinding is done as follows. Given a ciphertext c, instead of directly raising c to the d-th power, one generates a random value r, raises $c \cdot r^e$ to the d-th power, and multiplies the result by r^{-1}. Since $r^{ed} \cdot r^{-1} \equiv 1 \pmod{n}$, the result is correct; yet the value sent to the modular exponentiation routine is randomized, which foils chosen-ciphertext attacks.

While the above countermeasures indeed foil the attacks presented in this paper, the key distinguishing attack is unaffected and still present in the latest versions of GnuPG; mitigating it in software, without a large overhead, remains an open problem.

Physical mitigation. Physical mitigation techniques include Faraday cages (against EM attacks), insulating enclosures (against chassis and touch attacks), and photoelectric decoupling or fiberoptic connections (against "far end of cable" attacks). However, inexpensive protection of consumer-grade PCs appears difficult, especially for the chassis channel. Indeed, substantial engineering attention and expense are already dedicated to the mitigation of conducted emanations, for the purpose of electromagnetic compatibility (EMC) compliance, yet the channel evidently persists; moreover, the common method for filtering conducted emanations on power supply lines is to use bypass capacitors to shunt stray AC currents into the ground, but this obviously does not apply to the ground line itself. Robust low-impedance grounding and shielding, with careful attention to current paths, should reduce voltages across the ground and chassis (at costs in engineering effort and portability). We conjecture that prudent design of switching power supplies and voltage regulators can reduce computation-dependent modulation of emanations without significantly hampering efficiency, and we raise this challenge as another open problem.

Acknowledgments

We are indebted to Adi Shamir for insightful discussions and suggestions, and to Lev Pachmanov for writing much of the software setup used in our experiments. Ezra Shaked assisted in constructing and configuring the experimental setup. Assa Naveh assisted in various experiments, and offered valuable suggestions. Sharon Kessler provided copious editorial advice.

This work was sponsored by the Check Point Institute for Information Security; by European Union's Tenth Framework Programme (FP10/2010-2016) under grant agreement no. 259426 ERC-CaC, by the the Leona M. & Harry B. Helmsley Charitable Trust; by the Israeli Ministry of Science and Technology; by the Israeli Centers of Research Excellence I-CORE program (center 4/11); and by NATO's Public Diplomacy Division in the Framework of "Science for Peace".

References

[AARR02] Dakshi Agrawal, Bruce Archambeault, Josyula R. Rao, and Pankaj Rohatgi. The EM side-channel(s). In *CHES*, pages 29–45, 2002.

[And08] Ross J. Anderson. *Security engineering — a guide to building dependable distributed systems (2nd ed.)*. Wiley, 2008.

[BB05] David Brumley and Dan Boneh. Remote timing attacks are practical. *Computer Networks*, 48(5):701–716, 2005.

[Ber05] Daniel J. Bernstein. Cache-timing attacks on AES. 2005. URL: `http://cr.yp.to/papers.html#cachetiming`.

[BT11] Billy Bob Brumley and Nicola Tuveri. Remote timing attacks are still practical. In *ESORICS*, pages 355–371, 2011.

[CDF+07] J. Callas, L. Donnerhacke, H. Finney, D. Shaw, and R. Thayer. OpenPGP message format. RFC 4880, November 2007.

[CFR10] Jean-Christophe Courrège, Benoit Feix, and Mylène Roussellet. Simple power analysis on exponentiation revisited. In *CARDIS*, pages 65–79, 2010.

[CMR+13] Shane S. Clark, Hossen A. Mustafa, Benjamin Ransford, Jacob Sorber, Kevin Fu, and Wenyuan Xu. Current events: Identifying webpages by tapping the electrical outlet. In *ESORICS*, pages 700–717, 2013.

[Cop97] Don Coppersmith. Small solutions to polynomial equations, and low exponent RSA vulnerabilities. *J. Cryptology*, 10(4):233–260, 1997.

[ElG85] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.

[Eni] The Enigmail Project. Enigmail: A simple interface for OpenPGP email security. URL: `https://www.enigmail.net`.

[ETLR01] M. Elkins, D. Del Torto, R. Levien, and T. Roessler. MIME security with OpenPGP. RFC 3156, August 2001. URL: `http://www.ietf.org/rfc/rfc3156.txt`.

[FV03] Pierre-Alain Fouque and Frédéric Valette. The doubling attack — why upwards is better than downwards. In *CHES*, pages 269–280, 2003.

[GMO01] Karine Gandolfi, Christophe Mourtel, and Francis Olivier. Electromagnetic analysis: concrete results. In *CHES*, pages 251–261, 2001.

[Gmp] GNU multiple precision arithmetic library. URL: `http://gmplib.org/`.

[Gpg] The GNU Privacy Guard. URL: `http://www.gnupg.org`.

[GST13] Daniel Genkin, Adi Shamir, and Eran Tromer. RSA key extraction via low-bandwidth acoustic cryptanalysis (extended version). *IACR Cryptology ePrint Archive*, 2013:857, 2013. Extended version of [GST14].

[GST14] Daniel Genkin, Adi Shamir, and Eran Tromer. RSA key extraction via low-bandwidth acoustic cryptanalysis. In *CRYPTO*, pages 444–461 (vol. 1), 2014. See [GST13] for extended version.

[HMA+08] Naofumi Homma, Atsushi Miyamoto, Takafumi Aoki, Akashi Satoh, and Adi Shamir. Collision-based power analysis of modular exponentiation using chosen-message pairs. In *CHES*, pages 15–29, 2008.

[Hu92] Wei-Ming Hu. Lattice scheduling and covert channels. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 52–61, 1992.

[KJJ99] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. pages 388–397. Springer, 1999.

[KJJR11] Paul Kocher, Joshua Jaffe, Benjamin Jun, and Pankaj Rohatgi. Introduction to differential power analysis. *Journal of Cryptographic Engineering*, 1(1):5–27, 2011.

[KO62] A. Karatsuba and Y. Ofman. Multiplication of Many-Digital Numbers by Automatic Computers. *Proceedings of the USSR Academy of Sciences*, 145:293–294, 1962.

[Koc96] Paul C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. pages 104–113. Springer, 1996.

[Kuh03] Markus G. Kuhn. Compromising emanations: Eavesdropping risks of computer displays, 2003. PhD dissertation.

[MDS99] Thomas S. Messerges, Ezzy A. Dabbish, and Robert H. Sloan. Power analysis attacks of modular exponentiation in smartcards. In *CHES*, pages 144–157, 1999.

[MIT13] MITRE. Common vulnerabilities and exposures list, entry CVE-2013-4576, 2013. URL: `http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2013-4576`.

[MOP07] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power analysis attacks — revealing the secrets of smart cards*. Springer, 2007.

[Nov02] Roman Novak. SPA-based adaptive chosen-ciphertext attack on RSA implementation. In *Public Key Cryptography*, pages 252–262, 2002.

[OS06] Yossi Oren and Adi Shamir. How not to protect PCs from power analysis, 2006. CRYPTO rump session. URL: `http://iss.oy.ne.ro/HowNotToProtectPCsFromPowerAnalysis`.

[OST06] Dag Arne Osvik, Adi Shamir, and Eran Tromer. Cache attacks and countermeasures: The case of AES. In *CT-RSA*, pages 1–20, 2006.

[Per05] Colin Percival. Cache missing for fun and profit. Presented at BSDCan, 2005. URL: `http://www.daemonology.net/hyperthreading-considered-harmful`.

[QS01] Jean-Jacques Quisquater and David Samyde. Electromagnetic analysis (EMA): Measures and counter-measures for smart cards. In *E-smart'01*, pages 200–210, 2001.

[SPK+10] Jörn-Marc Schmidt, Thomas Plos, Mario Kirschbaum, Michael Hutter, Marcel Medwed, and Christoph Herbst. Side-channel leakage across borders. In *CARDIS*, pages 36–48, 2010.

[TB10] C. Tokunaga and D. Blaauw. Securing encryption systems with a switched capacitor current equalizer. *Solid-State Circuits, IEEE Journal of*, 45(1):23–31, Jan 2010.

[WS05] Colin D. Walter and David Samyde. Data dependent power use in multipliers. In *IEEE Symposium on Computer Arithmetic*, pages 4–12, 2005.

[WT01] Colin D. Walter and Susan Thompson. Distinguishing exponent digits by observing modular subtractions. In *CT-RSA*, pages 192–207, 2001.

[YF13] Yuval Yarom and Katrina E. Falkner. Flush+reload: a high resolution, low noise, L3 cache side-channel attack. *IACR Cryptology ePrint Archive*, 2013:448, 2013.

[YLMH05] Sung-Ming Yen, Wei-Chih Lien, Sang-Jae Moon, and JaeCheol Ha. Power analysis by exploiting chosen message and internal collisions — vulnerability of checking mechanism for RSA-decryption. In *Mycrypt*, pages 183–195, 2005.

[ZP14] A. Zajic and M. Prvulovic. Experimental demonstration of electromagnetic information leakage from modern processor-memory systems. *IEEE Transactions on Electromagnetic Compatibility*, 56(4):885–893, 2014.