DOI:10.1145/2851486

Computers broadcast their secrets via inadvertent physical emanations that are easily measured and exploited.

BY DANIEL GENKIN, LEV PACHMANOV, ITAMAR PIPMAN, ADI SHAMIR, AND ERAN TROMER

Physical Key Extraction Attacks on PCs

CRYPTOGRAPHY IS UBIQUITOUS. Secure websites and financial, personal communication, corporate, and national secrets all depend on cryptographic algorithms operating correctly. Builders of cryptographic systems have learned (often the hard way) to devise algorithms and protocols with sound theoretical analysis, write software that implements them correctly, and robustly integrate them with the surrounding applications. Consequentially, direct attacks against state-of-the-art cryptographic software are getting increasingly difficult.

For attackers, ramming the gates of cryptography is not the only option. They can instead undermine the fortification by violating basic assumptions made by the cryptographic software. One such assumption is software can control its outputs. Our programming courses explain that programs produce their outputs through designated interfaces (whether `print`, `write`, `send`, or `mmap`); so, to keep a secret, the software just needs to never output it or anything that may reveal it. (The operating system may be misused to allow someone else's process to peek into the program's memory or files, though we are getting better at avoiding such attacks, too.)

Yet programs' control over their own outputs is a convenient fiction, for a deeper reason. The hardware running the program is a physical object and, as such, interacts with its environment in complex ways, including electric currents, electromagnetic fields, sound, vibrations, and light emissions. All these "side channels" may depend on the computation performed, along with the secrets within it. "Side-channel attacks," which exploit such information leakage, have been used to break the security of numerous cryptographic implementations; see Anderson,[2] Kocher et al.,[19] and Mangard et al.[23] and references therein.

Side channels on small devices. Many past works addressed leakage from small devices (such as smartcards, RFID tags, FPGAs, and simple embedded devices); for such devices, physical key extraction attacks have been demonstrated with devastating effectiveness and across multiple physical channels. For example, a device's power consumption is often correlated with the computation it is currently executing. Over the past two decades, this physical phenomenon has been used extensively for key extraction from small devices,[19,23] often using powerful techniques, including differential power analysis.[18]

» key insights

- Small differences in a program's data can cause large differences in acoustic, electric, and electromagnetic emanations as the program runs.
- These emanations can be measured through inexpensive equipment and used to extract secret data, even from fast and complex devices like laptop computers and mobile phones.
- Common hardware and software are vulnerable, and practical mitigation of these risks requires careful application-specific engineering and evaluation.

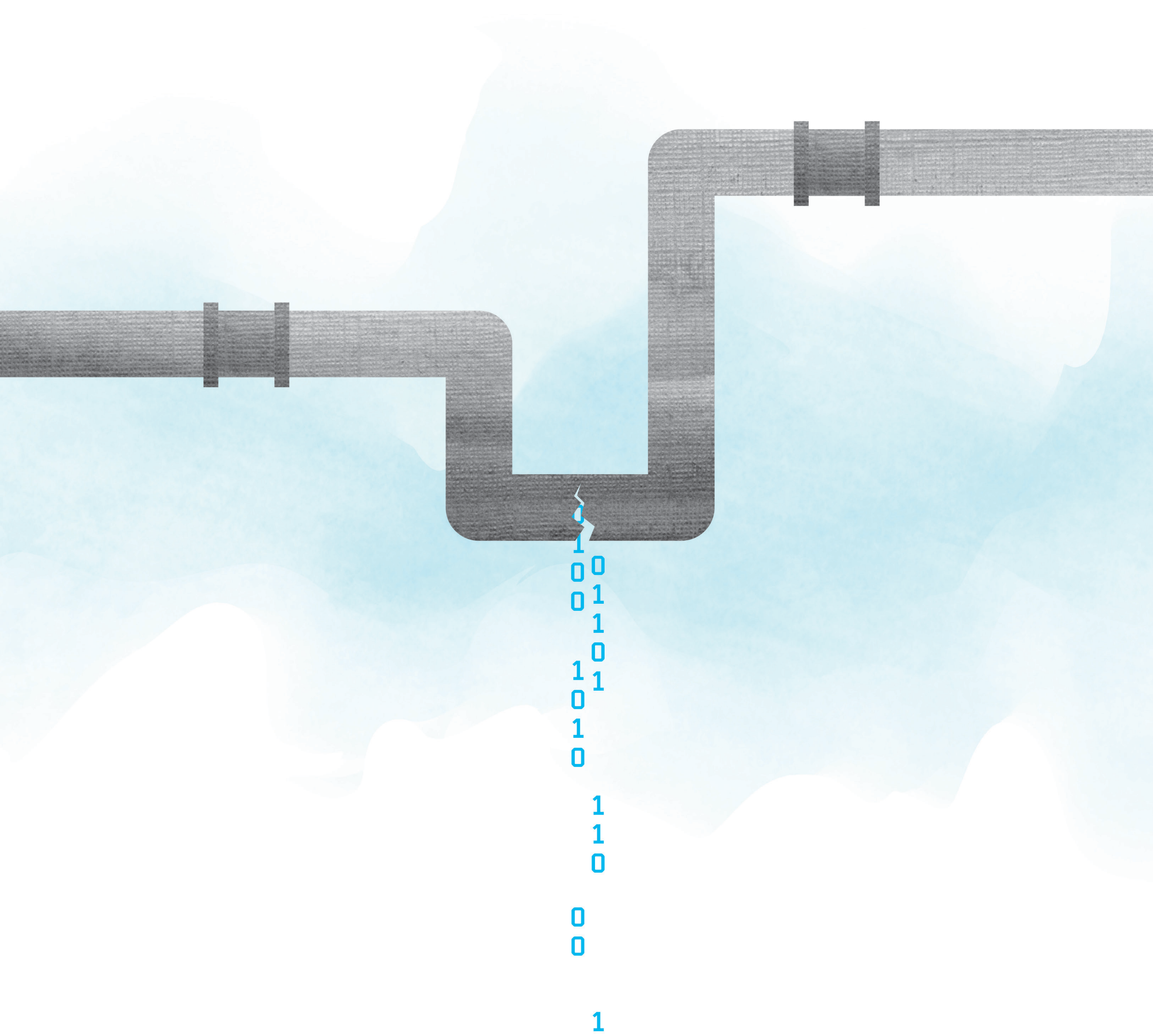


IMAGE BY IWONA USAKIEWICZ/ANDRIJ BORYS ASSOCIATES

The electromagnetic emanations from a device are likewise affected by the computation-correlated currents inside it. Starting with Agrawal et al.,[1] Gandolfi et al.,[11] and Quisquater and Samyde,[28] such attacks have been demonstrated on numerous small devices involving various cryptographic implementations.

Optical and thermal imaging of circuits provides layout information and coarse activity maps that are useful for reverse engineering. Miniature probes can be used to access individual internal wires in a chip, though such techniques require invasive disassembly of the chip package, as well as considerable technical expertise. Optical emanations from transistors, as they switch state, are exploitable as a side channel for reading internal registers leading and extracting keys.[29]

See Anderson[2] for an extensive survey of such attacks.

Vulnerability of PCs. Little was known, however, about the possibility of cryptographic attacks through physical side channels on modern commodity laptop, desktop, and server computers. Such "PC-class" computers (or "PCs," as we call them here) are indeed very different from the aforementioned small devices, for several reasons.

First, a PC is a very complex environment—a CPU with perhaps one billion transistors, on a motherboard with other circuitry and peripherals, running an operating system and handling various asynchronous events. All these introduce complexity, unpredictability, and noise into the physical emanations as the cryptographic code executes.

Second is speed. Typical side-channel techniques require the analog leakage signal be acquired at a bandwidth greater than the target's clock rate. For PCs running GHz-scale CPUs, this means recording analog signals at multi-GHz bandwidths requiring expensive and delicate lab equipment, in addition to a lot of storage space and processing power.

Figure 1. An acoustic attack using a parabolic microphone (left) on a target laptop (right); keys can be extracted from a distance of 10 meters.

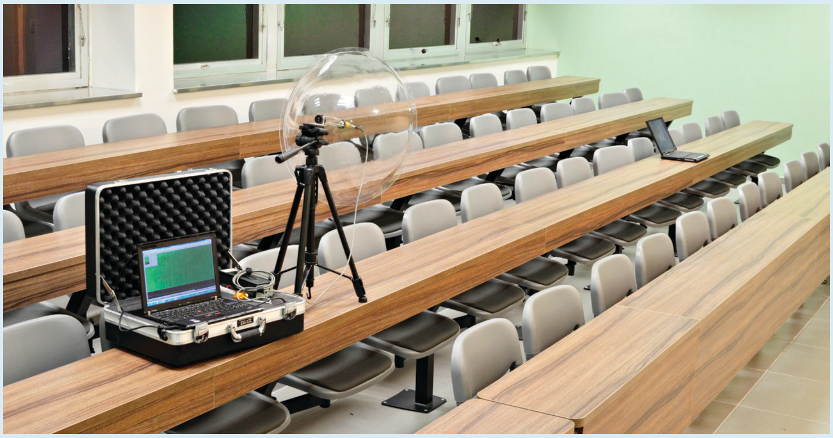
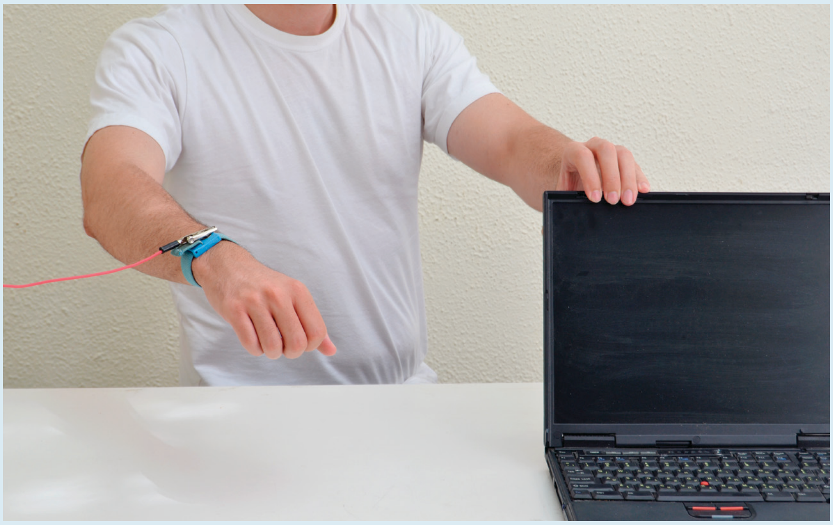


Figure 2. Measuring the chassis potential by touching a conductive part of the laptop; the wristband is connected to signal-acquisition equipment.



A third difference involves attack scenarios. Traditional techniques for side-channel attacks require long, uninterrupted physical access to the target device. Moreover, some such attacks involve destructive mechanical intrusion into the device (such as decapsulating chips). For small devices, these scenarios make sense; such devices are often easily stolen and sometimes even handed out to the attacker (such as in the form of cable TV subscription cards). However, when attacking other people's PCs, the attacker's physical access is often brief, constrained, and must proceed unobserved.

Note numerous side channels in PCs are known at the software level; timing,[8] cache contention,[6,26,27] and many other effects can be used to glean sensitive information across the boundaries between processes or even virtual machines. Here, we focus on physical attacks that do not require deployment of malicious software on the target PC.

Our research thus focuses on two main questions: Can physical side-channel attacks be used to nonintrusively extract secret keys from PCs, despite their complexity and operating speed? And what is the cost of such attacks in time, equipment, expertise, and physical access?

Results. We have identified multiple side channels for mounting physical key-extraction attacks on PCs, applicable in various scenarios and offering various trade-offs among attack range, speed, and equipment cost. The following sections explore our findings, as published in several recent articles.[12,15,16]

Acoustic. The power consumption of a CPU and related chips changes drastically (by many Watts) depending on the computation being performed at each moment. Electronic components in a PC's internal power supply, struggling to provide constant voltage to the chips, are subject to mechanical forces due to fluctuations of voltages and currents. The resulting vibrations, as transmitted to the ambient air, create high-pitched acoustic noise, known as "coil whine," even though it often originates from capacitors. Because this noise is correlated with the ongoing computation, it leaks information about what applications are running and what data they process. Most dramatically, it can acoustically leak secret keys during cryptographic operations.

By recording such noise while a target is using the RSA algorithm to decrypt ciphertexts (sent to it by the attacker), the RSA secret key can be extracted within one hour for a high-grade 4,096-bit RSA key. We experimentally demonstrated this attack from as far as 10 meters away using a parabolic microphone (see Figure 1) or from 30cm away through a plain mobile phone placed next to the computer.

Electric. While PCs are typically grounded to the mains earth (through their power supply "brick," or grounded peripherals), these connections are, in practice, not ideal, so the electric potential of the laptop's chassis fluctuates. These fluctuations depend on internal currents, and thus on the ongoing computation. An attacker can measure the fluctuations directly through a plain wire connected to a conductive part of the laptop, or indirectly through any cable with a conductive shield attached to an I/O port on the laptop (such as USB, Ethernet, display, or audio). Perhaps most surprising, the chassis potential can be measured, with sufficient fidelity, even through a human body; human attackers need to touch only the target computer with a bare hand while their body potential is measured (see Figure 2).

This channel offers a higher bandwidth than the acoustic one, allowing observation of the effect of individual key bits on the computation. RSA and ElGamal keys can thus be extracted from a signal obtained from just a few seconds of measurement, by touching a conductive part of the laptop's chassis, or by measuring the chassis potential from the far side of a 10-meter-long cable connected to the target's I/O port.

Electromagnetic. The computation performed by a PC also affects the electromagnetic field it radiates. By monitoring the computation-dependent electromagnetic fluctuations through an antenna for just a few seconds, it is possible to extract RSA and ElGamal secret keys. For this channel, the measurement setup is notably unintrusive and simple. A suitable electromagnetic probe antenna can be made from a simple loop of wire and recorded through an inexpensive software-defined radio USB dongle. Alternatively, an attacker can sometimes use a plain consumer-grade AM radio receiver, tuned close to the target's signal frequency, with its headphone output connected to a phone's audio jack for digital recording (see Figure 3).

Applicability. A surprising result of our research is how practical and easy are physical key-extraction side-channel attacks on PC-class devices, despite the devices' apparent complexity and high speed. Moreover, unlike previous attacks, our attacks require very little analog bandwidth, as low as 50kHz, even when attacking multi-GHz CPUs, thus allowing us to utilize new channels, as well as inexpensive and readily available hardware.

We have demonstrated the feasibility of our attacks using GnuPG (also known as GPG), a popular open source cryptographic software that implements both RSA and ElGamal. Our attacks are effective against various versions of GnuPG that use different implementations of the targeted cryptographic algorithm. We tested various laptop computers of different models from different manufacturers and running various operating systems, all "as is," with no modification or case intrusions.

History. Physical side-channel attacks have been studied for decades in military and espionage contexts in the U.S. and NATO under the codename TEMPEST. Most of this work remains classified. What little is declassified confirms the existence and risk of physical information leakage but says nothing about the feasibility of the key extraction scenarios discussed in this article. Acoustic leakage, in particular, has been used against electromechanical ciphers (Wright[31] recounts how the British security agencies tapped a phone to eavesdrop on the rotors of a Hagelin electromechanical cipher machine); but there is strong evidence it was not recognized by the security services as effective against modern electronic computers.[16]

Non-Cryptographic Leakage

Peripheral devices attached to PCs are prone to side-channel leakage due to their physical nature and lower operating speed; for example, acoustic noise from keyboards can reveal keystrokes,[3] printer-noise printed content,[4] and status LEDs data on a communication line.[22] Computer screens inadvertently broadcast their content as "van Eck" electromagnetic radiation that can be picked up from a distance;[21,30] see Anderson[2] for a survey.

Some observations have also been made about physical leakage from PCs, though at a coarse level. The general activity level is easily gleaned from temperature,[7] fan speed, and mechanical hard-disk movement. By tapping the computer's electric AC power, it is possible to identify the webpages

Figure 3. An electromagnetic attack using a consumer AM radio receiver placed near the target and recorded by a smartphone.

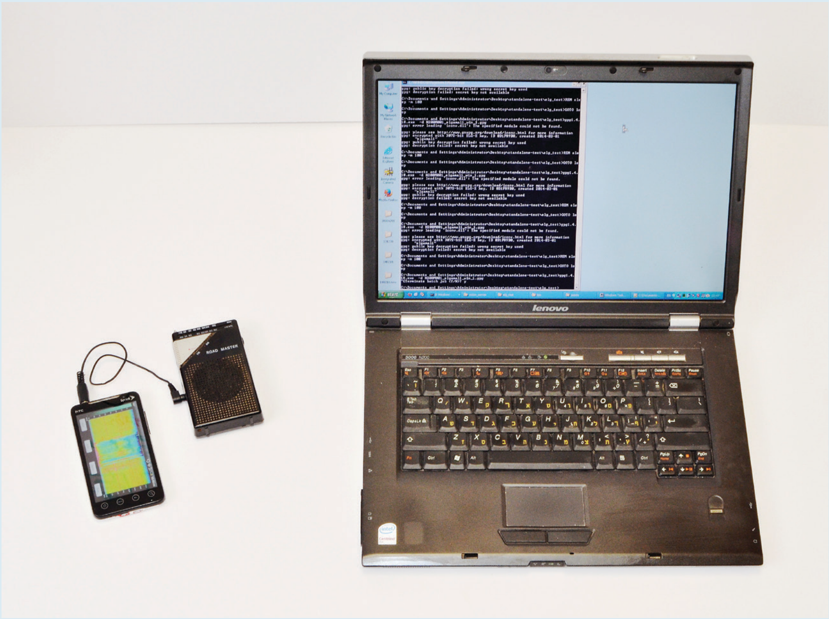
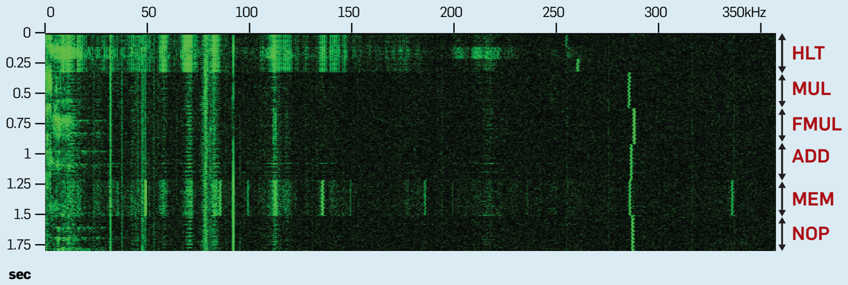


Figure 4. A spectrogram of an acoustic signal. The vertical axis is time (3.7 seconds), and the horizontal axis is frequency (0kHz–310kHz). Intensity represents instantaneous energy in the frequency band. The target is performing one-second loops of several x86 instructions: CPU sleep (`HLT`), integer multiplication (`MUL`), floating-point multiplication (`FMUL`), main memory access, and short-term idle (`REP NOP`).



loaded by the target's browser[9] and even some malware.[10] Tapping USB power lines makes it possible to identify when cryptographic applications are running.[25]

The acoustic, electric, and electromagnetic channels can also be used to gather coarse information about a target's computations; Figure 4 shows a microphone recording of a PC, demonstrating loops of different operations have distinct acoustic signatures.

Cryptanalytic Approach

Coarse leakage is ubiquitous and easily demonstrated once the existence of the physical channel is recognized. However, there remains the question of whether the physical channels can be used to steal finer and more devastating information. The crown jewels, in this respect, are cryptographic keys, for three reasons. First, direct impact, as compromising cryptographic keys endangers all data and authorizations that depend on them. Second, difficulty, as cryptographic keys tend to be well protected and used in carefully crafted algorithms designed to resist attacks; so if even these keys can be extracted, it is a strong indication more pedestrian data can be also extracted. And third, commonality, as there is only a small number of popular cryptographic algorithms and implementations, so compromising any of them has a direct effect on many deployed systems. Consequently, our research focused on key extraction from the most common public-key encryption schemes—RSA and ElGamal—as implemented by the popular GnuPG software.

When analyzing implementations of public-key cryptographic algorithms, an attacker faces the difficulties described earlier of complexity, noise, speed, and nonintrusiveness. Moreover, engineers implementing cryptographic algorithms try to make the sequence of executed operations very regular and similar for all secret keys. This is done to foil past attacks that exploit significant changes in control flow to deduce secrets, including timing attacks,[8] cache contention attacks[6,26,27] (such as a recent application to GnuPG[32,33]), and many other types of attacks on small devices.

We now show how to overcome these difficulties, using a careful selection of the ciphertext to be decrypted by the algorithm. By combining the following two techniques for ciphertext selection, we obtain a key-dependent leakage that is robustly observable, even through low-bandwidth measurements.

Internal value poisoning. While the sequence of performed operations is often decoupled from the secret key, the operands to these operations are often key-dependent. Moreover, operand values with atypical properties (such as operands containing many zero bits or that are unusually short) may trigger implementation-dependent corner cases. We thus craft special inputs (ciphertexts to be decrypted) that "poison" internal values occurring inside the cryptographic algorithm, so atypically structured operands occur at key-dependent times. Measuring leakage during such a poisoned execution can reveal at which operations these operands occurred, and thus leak key information.

Leakage self-amplification. In order to overcome a device's complexity and execution speed, an attacker can exploit the algorithm's own code to amplify its own leakage. By asking for decryption of a carefully chosen ciphertext, we create a minute change (compared to the decryption of a random-looking ciphertext) during execution of the innermost loop of the attacked algorithm. Since the code inside the innermost loop is executed many times throughout the algorithm, this yields an easily observable global change affecting the algorithm's entire execution.

Algorithm 1. Modular exponentiation using square-and-always-multiply.

```
Input: Three integers $c,d,q$ in binary representation such
   that $d = d_1 \cdots d_m$.
Output: $a = c^d$ mod $q$.
procedure MOD_EXP($c,d,q$)
   $c \leftarrow c$ mod $q$
   $a \leftarrow 1$
   for $i \leftarrow 1$ to $m$ do
      $a \leftarrow a^2$
      $t \leftarrow a \cdot c$
      if $d_i = 1$ then
         $a \leftarrow t$
   return $a$
```

Algorithm 2. GnuPG's basic multiplication code.

```
Input: Two integers $a = a_s \cdots a_1$ and $b = b_t \cdots b_1$ of size $s$.
   and $t$ limbs respectively
Output: $a \cdot b$.
procedure MUL_BASECASE($a,b$)
   $p \leftarrow a \cdot b_1$
   for $i \leftarrow 2$ to $t$ do
      if $b_i \neq 0$ then      ▷ (and if $b_i = 0$ do nothing)
         $p \leftarrow p + a \cdot b_i \cdot 2^{32 \cdot (i-1)}$
   return $p$
```

GnuPG'S RSA Implementation

For concreteness in describing our basic attack method, we outline GnuPG's implementation of RSA decryption, as of version 1.4.14 from 2013. Later GnuPG versions revised their implementations to defend against the adaptive attack described here; we discuss these variants and corresponding attacks later in the article.

Notation. RSA key generation is done by choosing two large primes p, q, a public exponent e and a secret exponent d, such that $ed \equiv 1 \ (mod\ \Phi(n))$ where $n = pq$ and $\Phi(n) = (p-1)(q-1)$. The public key is (n, e) and the private key is (p, q, d). RSA encryption of a message m is done by computing m^e mod n, and RSA decryption of a ciphertext c is done by computing c^d *mod* n. GnuPG uses a common optimization for RSA decryption; instead of directly computing $m = c^d$ mod n, it first computes $m_p = c^{d_p}$ *mod* p, $m_q = c^{d_q}$ mod q (where d_p and d_q are derived from the secret key), then combines m_p and m_q

into m using the Chinese Remainder Theorem. To fully recover the secret key, it suffices to learn any of its components (p, q, d, dp, or d_q); the rest can be deduced.

Square-and-always-multiply exponentiation. Algorithm 1 is pseudocode of the square-and-always-multiply exponentiation used by GnuPG 1.4.14 to compute m_p and m_q. As a countermeasure to the attack of Yarom and Falkner,[32] the sequence of squarings and multiplications performed by Algorithm 1 is independent of the secret key. Note the modular reduction in line 2 and the multiplication in line 6. Both these lines are used by our attack on RSA—line 2 for poisoning internal values and line 6 for leakage self-amplification.

Since our attack uses GnuPG's multiplication routine for leakage self-amplification, we now analyze the code of GnuPG's multiplication routines.

Multiplication. For multiplying large integers (line 6), GnuPG uses a variant of the Karatsuba multiplication algorithm. It computes the product of two k-numbers a and b recursively, using the identity $ab = (2^{2k} + 2^k)a_Hb_H + 2^k(a_H - a_L)(b_L - b_L) + (2^k + 1)a_Lb_L$, where a_H, b_H are the most significant halves of a and b, respectively, and, similarly, a_L, b_L are the least significant halves of a and b.

The recursion's base case is a simple grade-school "long multiplication" algorithm, shown (in simplified form) in Algorithm 2. GnuPG stores large integers in arrays of 32-bit words, called limbs. Note how Algorithm 2 handles the case of zero limbs of b. Whenever a zero limb of b is encountered, the operation in line 5 is not executed, and the loop in line 3 proceeds to handle the next limb of b. This optimization is exploited by the leakage self-amplification component of our attack. Specifically, each of our chosen ciphertexts will cause a targeted bit of q to affect the number of zero limbs of b given to Algorithm 2 and thus the control flow in line 4 and thereby the side-channel leakage.

Adaptive Chosen Ciphertext Attack

We now describe our first attack on RSA, extracting the bits of the secret prime q, one by one. For each bit of q, denoted q_i, the attack chooses a ciphertext $c^{(i)}$ such that when $c^{(i)}$ is decrypted by the target the side-channel leakage reveals the value of q_i. Eventually the entire q is revealed. The choice of each ciphertext depends on the key bits learned thus far, making it an adaptive chosen ciphertext attack.

This attack requires the target to decrypt ciphertexts chosen by the attacker, which is realistic since GnuPG is invoked by numerous applications to decrypt ciphertexts arriving via email messages, files, webpages, and chat messages. For example, Enigmail and GpgOL are popular plugins that add PGP/MIME encrypted-email capabilities to Mozilla Thunderbird and Outlook, respectively. They decrypt incoming email messages by passing them to GnuPG. If the target uses them, an attacker can remotely inject a chosen ciphertext into GnuPG by encoding the ciphertext as a PGP/MIME email (following RFC 3156) and sending it to the target.

Cryptanalysis. We can now describe the adaptive chosen ciphertext attack on GnuPG's RSA implementation.

Internal value poisoning. We begin by choosing appropriate ciphertexts that will poison some of the internal values inside Algorithm 1. Let p, q be two random k-bit primes comprising an RSA secret key; in the case of high-security 4,096-bit RSA, $k = 2{,}048$. GnuPG always generates RSA keys such that the most significant bit of p and q is set, thus $q_i = 1$. Assume we have already recovered the topmost $i - 1$ bits of q and define the ciphertext $c^{(i)}$ to be the k-bit ciphertext whose topmost $i - 1$ bits are the same as q, its i-th bit is 0 and whose remaining bits are set to 1. Consider the effects of decrypting $c^{(i)}$ on the intermediate values of Algorithm 1, depending on the secret key bit q_i.

Suppose $q_i = 1$. Then $c^{(i)} \leq q$, and this $c^{(i)}$ is passed as the argument c to Algorithm 1, where the modular reduction in line 2 returns $c = c^{(i)}$ (since $c^{(i)} \leq q$), so the lowest $k - i$ bits of c remain 1. Conversely, if $q_i = 0$, then $c^{(i)} > q$, so when $c^{(i)}$ is passed to Algorithm 1, the modular reduction in line 2 modifies the value of c. Since $c^{(i)}$ agrees with q on its topmost $i - 1$ bits, it holds that $q < c^{(i)} < 2q$, so in this case the modular reduction computes $c \leftarrow c - q$, which is a random-looking number of length $k - i$ bits.

We have thus obtained a connection between the i-th bit of q and the resulting structure of c after the modular reduction—either long and repetitive or short and random looking—thereby poisoning internal values in Algorithm 1.

Leakage self-amplification. To learn the i-th bit of q, we need to amplify the leakage resulting from this connection so it becomes physically distinguishable. Note the value c is used during the main loop of Algorithm 1 in line 6. Moreover, since the multiplication in line 6 is executed once per bit of d, we obtain that Algorithm 1 performs k multiplications by c, whose structure depends on q_i. We now analyze the effects of repetitive vs. random-looking second operand on the multiplication routine of GnuPG.

Suppose $c^{(i)} = 1$. Then c has its lowest $k - i$ bits set to 1. Next, c is passed to the Karatsuba-based multiplication routine as the second operand b. The result of $(b_L - b_H)$, as computed in the Karatsuba-based multiplication, will thus contain many zero limbs. This invariant, of having the second operand containing many zero limbs, is preserved by the Karatsuba-based multiplication all the way until the recursion reaches the base-case multiplication routine (Algorithm 2), where it affects the control flow in line 4, forcing the loop in line 3 to perform almost no multiplications.

Conversely, if $q_i = 0$, then c is random-looking, containing few (if any) zero limbs. When the Karatsuba-based multiplication routine gets c as its second operand b, the derived values stay random-looking throughout the recursion until the base case, where these random-looking values affect the control flow in line 4 inside the main loop of Algorithm 2, making it almost always perform a multiplication.

Our attack thus creates a situation where, during the entire decryption operation, the branch in line 4 of Algorithm 2 is either always taken or is never taken, depending on the current bit of q. During the decryption process, the branch in line 4 is evaluated numerous times (approximately 129,000 times for 4,096-bit RSA). This yields the desired self-amplification effect. Once q_i is extracted, we can compute the next chosen ciphertext c^{i+1} and proceed to ex-

Figure 5. Measuring acoustic leakage: (a) is the attacked target; (b) is a microphone picking up the acoustic emanations; (c) is the microphone power supply and amplifier; (d) is the digitizer; and the acquired signal is processed and displayed by the attacker's laptop (e).

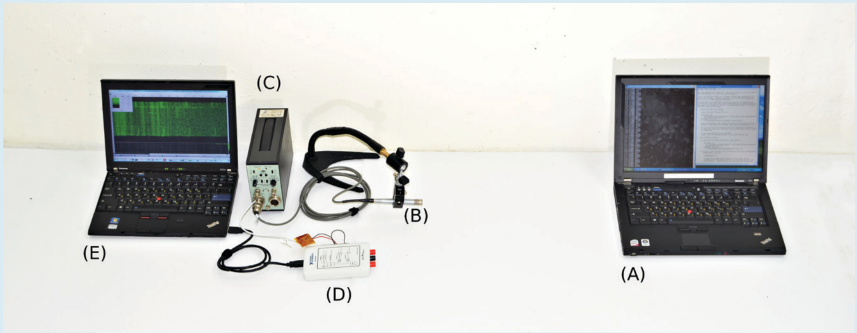
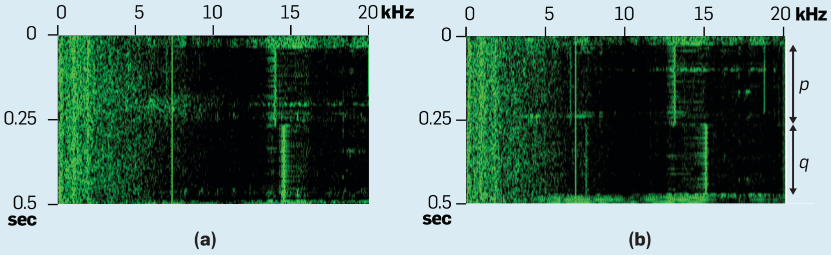


Figure 6. Acoustic emanations (0kHz–20kHz, 0.5 seconds) of RSA decryption during an adaptive chosen-ciphertext attack.



tract the next secret bit—q_{i+1}—through the same method.

The full attack requires additional components (such as error detection and recovery[16]).

Acoustic cryptanalysis of RSA. The basic experimental setup for measuring acoustic leakage consists of a microphone for converting mechanical air vibrations to electronic signals, an amplifier for amplifying the microphone's signals, a digitizer for converting the analog signal to a digital form, and software to perform signal processing and cryptanalytic deduction. Figure 1 and Figure 5 show examples of such setups using sensitive ultrasound microphones. In some cases, it even suffices to record the target through the built-in microphone of a mobile phone placed in proximity to the target and running the attacker's mobile app.[16]

Figure 6 shows the results of applying the acoustic attack for different values (0 or 1) of the attacked bit of q. Several effects are discernible. First, the transition between the two modular exponentiations (using the modulus p and q) is clearly visible. Second, note the acoustic signatures of the second exponentiation is different between Figure 6a and Figure 6b. This is exactly the effect created by our attack, which can be utilized to extract the bits of q.

By applying the iterative attack algorithm described earlier, attacking each key bit at a time by sending the chosen ciphertext for decryption and learning the key bit from the measured acoustic signal, the attacker can fully extract the secret key. For 4,096-bit RSA keys (which, according to NIST recommendations, should remain secure for decades), key extraction takes approximately one hour.

Parallel load. This attack assumes decryption is triggered on an otherwise-idle target machine. If additional software is running concurrently, then the signal will be affected, but the attack may still be feasible. In particular, if other software is executed through timeslicing, then the irrelevant timeslices can be identified and discarded. If other, sufficiently homogenous software is executed on a different core, then (empirically) the signal of interest is merely shifted. Characterizing the general case is an open question, but we conjecture that exploitable correlations will persist.

Non-Adaptive Chosen Ciphertext Attacks

The attack described thus far requires decryption of a new adaptively chosen ciphertext for every bit of the secret key, forcing the attacker to interact with the target computer for a long time (approximately one hour). To reduce the attack time, we turn to the electrical and electromagnetic channels, which offer greater analog bandwidth, though still orders of magnitude less than the target's CPU frequency. This increase in bandwidth allows the attacker to observe finer details about the operations performed by the target algorithm, thus requiring less leakage amplification.

Utilizing the increased bandwidth, our next attack trades away some of the leakage amplification in favor of reducing the number of ciphertexts. This reduction shortens the key-extraction time to seconds and, moreover, makes the attack non-adaptive, meaning the chosen ciphertexts can be sent to the target all at once (such as on a CD with a few encrypted files).

Cryptanalysis. The non-adaptive chosen ciphertext attack against square-and-always-multiply exponentiation (Algorithm 1) follows the approach of Yen et al.,[34] extracting the bits of d instead of q.

Internal value poisoning. Consider the RSA decryption of $c = n - 1$. As in the previous acoustic attack, c is passed to Algorithm 1, except this time, after the modular reduction in line 2, it holds that $c \equiv -1 \pmod{q}$. We now examine the effect of c on the squaring operation performed during the main loop of Algorithm 1.

First note the value of a during the execution of Algorithm 1 is always either 1 or −1 modulo q. Next, since $-1^2 \equiv 1^2 \equiv 1 \pmod{q}$, we have that the value of a in line 6 is always 1 modulo q. We thus obtain the following connection between the secret key bit d_{i-1} and the value of a at the start of the i-th iteration of Algorithm 1's main loop.

Suppose $d_{i-1} = 0$, so the branch in line 7 is not taken, making the value of a at the start of the i-th iteration be $1 \bmod q = 1$. Since GnuPG's internal representation does not truncate leading zeros, a contains many leading zero limbs that are then passed to the squaring routine during the i-th iteration. Conversely, if $d_{i-1} = 1$, then the branch in line 7 is taken, making the value of a at the start of the i-th iteration be −1 modulo q, represented as $p - 1$. Since q is a randomly generated prime, the value of a, and therefore the value sent to the squaring routine during the i-th iteration, is unlikely to contain any zero limbs.

We have thus poisoned some of the internal values of Algorithm 1, creating a connection between the bits of d and the intermediate values of a during the exponentiation.

Amplification. GnuPG's squaring routines are implemented in ways similar to the multiplication routines, including the optimizations for handling zero limbs, yielding leakage self-amplification, as in an adaptive attack.

Since each iteration of the exponentiation's main loop leaks one bit of the secret d, all the bits d can be extracted from (ideally) a single decryption of a single ciphertext. In practice, a few measurements are needed to cope with noise, as discussed here.

Windowed exponentiation. Many RSA implementations, including GnuPG version 1.4.16 and newer, use an exponentiation algorithm that is faster than Algorithm 1. In such an implementation, the exponent d is split into blocks of m bits (typically $m = 5$), either contiguous blocks (in "fixed window" or "m-ary" exponentiation) or blocks separated by runs of zero bits (in "sliding-window" exponentiation). The main loop, instead of handling the exponent one bit at a time, handles a whole block at every iteration, by multiplying a by c^x, where x is the block's value. The values c^x are pre-computed and stored in a lookup table (for all m-bit values x).

An adaptation of these techniques also allows attacking windowed exponentiation.[12] In a nutshell, we focus on each possible m-bit value x, one at a time, and identify which blocks in the exponent d, that is, which iterations of the main loop, contain x. This is done by crafting a ciphertext c such that c^x mod q contains many zero limbs. Leakage amplification and measurement then work similarly to the acoustic and electric attacks described earlier. Once we identify where each x occurred, we aggregate these locations to deduce the full key d.

Electric attacks. As discussed earlier, the electrical potential on the chassis of laptop computers often fluctuates (in reference to the mains earth ground) in a computation-dependent way. In addition to measuring this potential directly using a plain wire connected to the laptop chassis, it is possible to measure the chassis potential from afar using the conductive shielding of any cable attached to one of the laptop's I/O ports (see Figure 7) or from nearby by touching an exposed metal part of the laptop's chassis, as in Figure 2.

To cope with noise, we measured the electric potential during a few (typically 10) decryption operations. Each recording was filtered and demodulated. We used frequency-demodulation since it produced best results compared to amplitude and phase demodulations. We then combined the recordings using correlation-based averaging, yielding a combined signal (see Figure 8). The successive bits of d can be deduced from this combined signal. Full key extraction, using non-adaptive electric measurements, requires only a few seconds of measurements, as opposed to an hour using the adaptive attack. We obtained similar results for ElGamal encryption; Genkin et al.[15] offer a complete discussion.

Electromagnetic attacks. The electromagnetic channel, which exploits computation-dependent fluctuations in the electromagnetic field surrounding the target, can also be used for key extraction. While this channel was previously used for attacks on small devices at very close proximity,[1,11,28] the PC class of devices was only recently considered by Zajic and Prulovic[35] (without cryptographic applications).

Measuring the target's electromagnetic emanations requires an antenna, electronics for filtering and amplification, analog-to-digital conversion, and software for signal processing and cryptanalytic deduction. Prior works (on small devices) typically used cumbersome and expensive lab-grade

Figure 7. Measuring the chassis potential from the far side of an Ethernet cable (blue) plugged into the target laptop (10 meters away) through an alligator clip leading to measurement equipment (green wire).

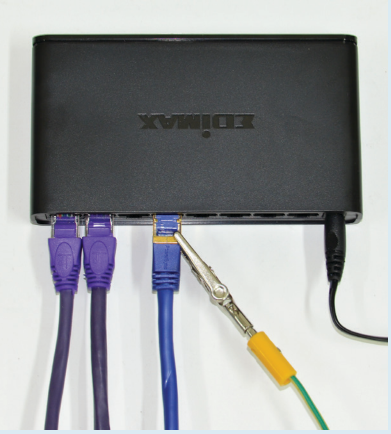
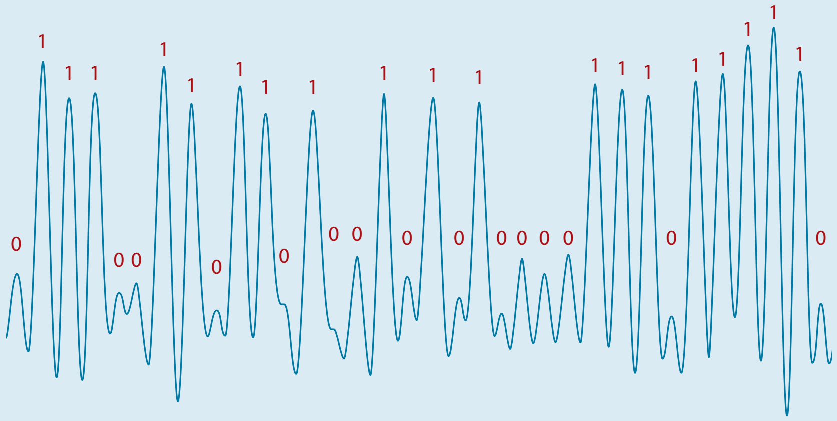


Figure 8. A signal segment from an electric attack, after demodulating and combining measurements of several decryptions. Note the correlation between the signal (blue) and the correct key bits (red).



equipment. In our attacks,[12] we used highly integrated solutions that are small and inexpensive (such as a software-defined radio dongle, as in Figure 9, or a consumer-grade radio receiver recorded by a smartphone, as in Figure 3). Demonstrating how an untethered probe may be constructed from readily available electronics, we also built the Portable Instrument for Trace Acquisition (PITA), which is compact enough to be concealed, as in pita bread (see Figure 10).

Experimental results. Attacking RSA and ElGamal (in both square-and-always-multiply and windowed implementations) over the electromagnetic channel (sampling at 200 kSample/sec around a center frequency of 1.7MHz), using the non-adaptive attack described earlier, we have extracted secret keys in a few seconds from a distance of half a meter.

Attacking other schemes and other devices. So far, we have discussed attacks on the RSA and ElGamal cryptosystems based on exponentiation in large prime fields. Similar attacks also target elliptic-curve cryptography. For example, we demonstrated key extraction from GnuPG's implementation of the Elliptic-Curve Diffie-Hellman scheme running on a PC;[13] the attacker, in this case, can measure the target's electromagnetic leakage from an adjacent room through a wall.

Turning to mobile phones and tablets, as well as to other cryptographic libraries (such as OpenSSL and iOS CommonCrypto), electromagnetic key extraction from implementations of the Elliptic Curve Digital Signature Algorithm has also been demonstrated, including attacks that are non-invasive,[17] low-bandwidth,[5,24] or both.[14]

Figure 9. Measuring electromagnetic emanations from a target laptop (left) through a loop of coax cable (handheld) recorded by a software-defined radio (right).

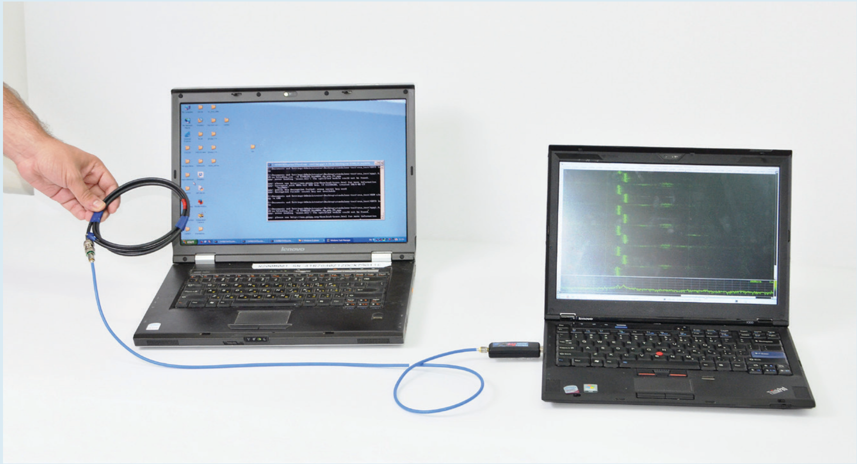
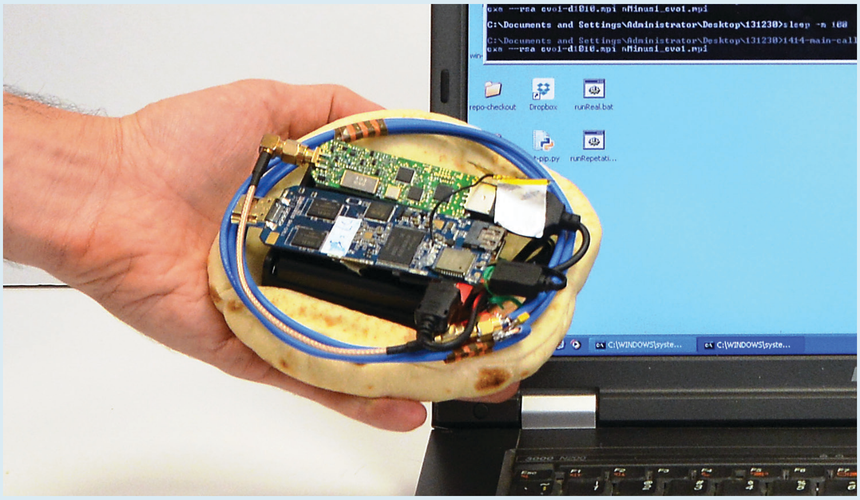


Figure 10. Extracting keys by measuring a laptop's electromagnetic emanations through a PITA device.



```
exe --rsa cvoi-d1010.mpi nMinus1_evo1.mpi

C:\Documents and Settings\Administrator\Desktop\131230>sleep -m 100

C:\Documents and Settings\Administrator\Desktop\131230>1414-main-call
exe --rsa cvoi-d1010.mpi nMinus1_evo1.mpi
```

Conclusion

Extraction of secret cryptographic keys from PCs using physical side channels is feasible, despite their complexity and execution speed. We have demonstrated such attacks on many public-key encryption schemes and digital-signature schemes, as implemented by popular cryptographic libraries, using inexpensive and readily available equipment, by various attack vectors and in multiple scenarios.

Hardware countermeasures. Side-channel leakage can be attenuated through such physical means as sound-absorbing enclosures against acoustic attacks, Faraday cages against electromagnetic attacks, insulating enclosures against chassis and touch attacks, and photoelectric decoupling or fiber-optic connections against "far end of cable" attacks. However, these countermeasures are expensive and cumbersome. Devising inexpensive physical leakage protection for consumer-grade PCs is an open problem.

Software countermeasures. Given a characterization of a side channel, algorithms and their software implementations may be designed so the leakage through the given channel will not convey useful information. One such approach is "blinding," or ensuring long operations (such as modular exponentiation) that involve sensitive values are, instead, performed on random dummy values and later corrected using an operation that includes the sensitive value but is much shorter and thus more difficult to measure (such as modular multiplication). A popular example of this approach is ciphertext randomization,[20] which was added to GnuPG following our observations and indeed prevents both the internal value poisoning and the leakage self-amplification components of our attacks.

However, such countermeasures require careful design and adaptation for every cryptographic scheme and leakage channel; moreover, they often involve significant cost in performance. There are emerging generic protection methods at the algorithmic level, using fully homomorphic encryption and cryptographic leakage resilience; however, their overhead is currently so great as to render them impractical.

Future work. To fully understand the ramifications and potential of physical side-channel attacks on PCs and other fast and complex devices, many questions remain open. What other implementations are vulnerable, and what other algorithms tend to have vulnerable implementations? In particular, can symmetric encryption algorithms (which are faster and more regular) be attacked? What other physical channels exist, and what signal processing and cryptanalytic techniques can exploit them? Can the attacks' range be extended (such as in acoustic attacks via laser vibrometers)? What level of threat do such channels pose in various real-world scenarios? Ongoing research indicates the risk extends well beyond the particular algorithms, software, and platforms we have covered here.

On the defensive side, we also raise three complementary questions: How can we formally model the feasible side-channel attacks on PCs? What engineering methods will ensure devices comply with the model? And what algorithms, when running on compliant devices, will provably protect their secrets, even in the presence of side-channel attacks?

Acknowledgments

This article is based on our previous research,[12,13,15,16] which was supported by the Check Point Institute for Information Security, the European Union's 10th Framework Programme (FP10/2010-2016) under grant agreement no. 259426 ERC-CaC, a Google Faculty Research Award, the Leona M. & Harry B. Helmsley Charitable Trust, the Israeli Ministry of Science, Technology and Space, the Israeli Centers of Research Excellence I-CORE program (center 4/11), NATO's Public Diplomacy Division in the Framework of "Science for Peace," and the Simons Foundation and DIMACS/Simons Collaboration in Cryptography through National Science Foundation grant #CNS-1523467.

References

1. Agrawal, D., Archambeault, B., Rao, J.R., and Rohatgi, P. The EM side-channel(s). In *Proceedings of the Workshop on Cryptographic Hardware and Embedded Systems (CHES 2002)*. Springer, 2002, 29–45.
2. Anderson, R.J. *Security Engineering: A Guide to Building Dependable Distributed Systems, Second Edition.* Wiley, 2008.
3. Asonov, D. and Agrawal, R. Keyboard acoustic emanations. In *Proceedings of the IEEE Symposium on Security and Privacy.* IEEE Computer Society Press, 2004, 3–11.
4. Backes, M., Dürmuth, M., Gerling, S., Pinkal, M., and Sporleder, C. Acoustic side-channel attacks on printers. In *Proceedings of the USENIX Security Symposium 2010.* USENIX Association, 2010, 307–322.
5. Belgarric, P., Fouque, P.-A., Macario-Rat, G., and Tibouchi, M. Side-channel analysis of Weierstrass and Koblitz curve ECDSA on Android smartphones. In *Proceedings of the Cryptographers' Track of the RSA Conference (CT-RSA 2016).* Springer, 2016, 236–252.
6. Bernstein, D.J. Cache-timing attacks on AES. 2005; http://cr.yp.to/papers.html#cachetiming
7. Brouchier, J., Dabbous, N., Kean, T., Marsh, C., and Naccache, D. *Thermocommunication.* Cryptology ePrint Archive, Report 2009/002, 2009; https://eprint.iacr.org/2009/002
8. Brumley, D. and Boneh, D. Remote timing attacks are practical. *Computer Networks 48*, 5 (Aug. 2005), 701–716.
9. Clark, S.S., Mustafa, H.A., Ransford, B., Sorber, J., Fu, K., and Xu, W. Current events: Identifying webpages by tapping the electrical outlet. In *Proceedings of the 18th European Symposium on Research in Computer Security (ESORICS 2013).* Springer, Berlin, Heidelberg, 2013, 700–717.
10. Clark, S.S., Ransford, B., Rahmati, A., Guineau, S., Sorber, J., Xu, W., and Fu, K. WattsUpDoc: Power side channels to nonintrusively discover untargeted malware on embedded medical devices. In *Proceedings of the USENIX Workshop on Health Information Technologies (HealthTech 2013).* USENIX Association, 2013.
11. Gandolfi, K., Mourtel, C., and Olivier, F. Electromagnetic analysis: Concrete results. In *Proceedings of the Workshop on Cryptographic Hardware and Embedded Systems (CHES 2001).* Springer, Berlin, Heidelberg, 2001, 251–261.
12. Genkin, D., Pachmanov, L., Pipman, I., and Tromer, E. Stealing keys from PCs using a radio: Cheap electromagnetic attacks on windowed exponentiation. In *Proceedings of the Workshop on Cryptographic Hardware and Embedded Systems (CHES 2015).* Springer, 2015, 207–228.
13. Genkin, D., Pachmanov, L., Pipman, I., and Tromer, E. ECDH key-extraction via low-bandwidth electromagnetic attacks on PCs. In *Proceedings of the Cryptographers' Track of the RSA Conference (CT-RSA 2016).* Springer, 2016, 219–235.
14. Genkin, D., Pachmanov, L., Pipman, I., Tromer, E., and Yarom, Y. *ECDSA Key Extraction from Mobile Devices via Nonintrusive Physical Side Channels.* Cryptology ePrint Archive, Report 2016/230, 2016; http://eprint.iacr.org/2016/230
15. Genkin, E., Pipman, I., and Tromer, E. Get your hands off my laptop: Physical side-channel key-extraction attacks on PCs. In *Proceedings of the Workshop on Cryptographic Hardware and Embedded Systems (CHES 2014).* Springer, 2014, 242–260.
16. Genkin, D., Shamir, A., and Tromer, E. RSA key extraction via low-bandwidth acoustic cryptanalysis. In *Proceedings of the Annual Cryptology Conference (CRYPTO 2014).* Springer, 2014, 444–461.
17. Kenworthy, G. and Rohatgi, P. Mobile device security: The case for side-channel resistance. In *Proceedings of the Mobile Security Technologies Conference (MoST)*, 2012; http://mostconf.org/2012/papers/21.pdf
18. Kocher, P., Jaffe, J., and Jun, B. Differential power analysis. In *Proceedings of the Annual Cryptology Conference (CRYPTO 1999).* Springer, 1999, 388–397.
19. Kocher, P., Jaffe, J., Jun, B., and Rohatgi, P. *Introduction to differential power analysis. Journal of Cryptographic Engineering 1*, 1 (2011), 5–27.
20. Kocher, P.C. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In *Proceedings of the Annual Cryptology Conference (CRYPTO 1996).* Springer, 1996, 104–113.
21. Kuhn, M.G. *Compromising Emanations: Eavesdropping Risks of Computer Displays.* Ph.D. Thesis and Technical Report UCAM-CL-TR-577. University of Cambridge Computer Laboratory, Cambridge, U.K., Dec. 2003; https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-577.pdf
22. Loughry, J. and Umphress, D.A. Information leakage from optical emanations. *ACM Transactions on Information Systems Security 5*, 3 (Aug. 2002), 262–289.
23. Mangard, S., Oswald, E., and Popp, T. *Power Analysis Attacks: Revealing the Secrets of Smart Cards.* Springer, Berlin, Heidelberg, 2007.
24. Nakano, Y., Souissi, Y., Nguyen, R., Sauvage, L., Danger, J., Guilley, S., Kiyomoto, S., and Miyake, Y. A pre-processing composition for secret key recovery on Android smartphones. In *Proceedings of the International Workshop on Information Security Theory and Practice (WISTP 2014).* Springer, Berlin, Heidelberg, 2014.
25. Oren, Y. and Shamir, A. How not to protect PCs from power analysis. Presented at the Annual Cryptology Conference (CRYPTO 2006) rump session. 2006; http://iss.oy.ne.ro/HowNotToProtectPCsFromPowerAnalysis
26. Osvik, D.A., Shamir, A., and Tromer, E. Cache attacks and countermeasures: The case of AES. In *Proceedings of the Cryptographers' Track of the RSA Conference (CT-RSA 2006)*. Springer, 2006,1–20.
27. Percival, C. Cache missing for fun and profit. In *Proceedings of the BSDCan Conference*, 2005; http://www.daemonology.net/hyperthreading-considered-harmful
28. Quisquater, J.-J. and Samyde, D. Electromagnetic analysis (EMA): Measures and countermeasures for smartcards. In *Proceedings of the Smart Card Programming and Security: International Conference on Research in Smart Cards (E-smart 2001).* Springer, 2001, 200–210.
29. Skorobogatov, S. *Optical Surveillance on Silicon Chips.* University of Cambridge, Cambridge, U.K., 2009; http://www.cl.cam.ac.uk/~sps32/SG_talk_OSSC_a.pdf
30. van Eck, W. Electromagnetic radiation from video display units: An eavesdropping risk? *Computers and Security 4*, 4 (Dec. 1985), 269–286.
31. Wright, P. *Spycatcher.* Viking Penguin, New York, 1987.
32. Yarom, Y. and Falkner, K. FLUSH+RELOAD: A high-resolution, low-noise, L3 cache side-channel attack. In *Proceedings of the USENIX Security Symposium 2014.* USENIX Association, 2014, 719–732.
33. Yarom, Y., Liu, F., Ge, Q., Heiser, G., and Lee, R.B. Last-level cache side-channel attacks are practical. In *Proceedings of the IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, 2015, 606–622.
34. Yen, S.-M., Lien, W.-C., Moon, S.-J., and Ha, J. Power analysis by exploiting chosen message and internal collisions: Vulnerability of checking mechanism for RSA decryption. In *Proceedings of the International Conference on Cryptology in Malaysia (Mycrypt 2005)*. Springer, 2005, 183–195.
35. Zajic, A. and Prvulovic, M. Experimental demonstration of electromagnetic information leakage from modern processor-memory systems. *IEEE Transactions on Electromagnetic Compatibility 56*, 4 (Aug. 2014), 885–893.

Daniel Genkin (danielg3@cs.technion.ac.il) is a Ph.D. candidate in the Computer Science Department at Technion-Israel Institute of Technology, Haifa, Israel, and a research assistant in the Blavatnik School of Computer Science at Tel Aviv University, Israel.

Lev Pachmanov (levp@tau.ac.il) is a master's candidate in the Blavatnik School of Computer Science at Tel Aviv University, Israel.

Itamar Pipman (itamarpi@tau.ac.il) is a master's candidate in the Blavatnik School of Computer Science at Tel Aviv University, Israel.

Adi Shamir (adi.shamir@weizmann.ac.il) is a professor in the faculty of Mathematics and Computer Science at the Weizmann Institute of Science, Rehovot, Israel.

Eran Tromer (tromer@cs.tau.ac.il) is a senior lecturer in the Blavatnik School of Computer Science at Tel Aviv University, Israel.