ECDH Key-Extraction via Low-Bandwidth Electromagnetic Attacks on PCs

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

Lev Pachmanov
Tel Aviv University
levp@post.tau.ac.il

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

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

February 11, 2016

Abstract

We present the first physical side-channel attack on elliptic curve cryptography running on a PC. The attack targets the ECDH public-key encryption algorithm, as implemented in the latest version of GnuPG. By measuring the target's electromagnetic emanations, the attack extracts the secret decryption key within seconds, from a target located in an adjacent room across a wall. The attack utilizes a single carefully chosen ciphertext, and tailored time-frequency signal analysis techniques, to achieve full key extraction.

1 Introduction

Physical side-channel attacks exploit unintentional information leakage via low-level physical behavior of computing devices, such as electromagnetic radiation, power consumption, electric potential, acoustic emanations and thermal fluctuations. These have been used to break numerous cryptographic implementations; see [And08, MOP07, KJJR11] and the references therein.

Small devices, such as smartcards, RFID tags, FPGAs, microcontrollers, and simple embedded devices, have received much research attention with numerous published side-channel attacks. However, for more complex "PC" class devices (laptops, dekstops, servers etc.), there are few physical side-channel attacks demonstrated on cryptographic implementations: key extraction from RSA using acoustic attacks [GST14], and key extraction from RSA and ElGamal using the ground-potential and electromagnetic channels [GPT14, GPPT15]. As discussed in those works, attacks on PCs raise new and difficult challenges compared to attacking small devices: hardware and software complexity causing unpredictable behavior and noise; high clock speeds of several GHz; and attack scenarios that force non-invasive attacks and limit signal quality, bandwidth and acquisition time. In particular, the effective measurement bandwidth is much lower than the target CPU's clock rate, making it infeasible to distinguish individual instructions and necessitating new, algorithm-specific cryptanalytic techniques.

This leaves open the question of what other cryptographic algorithm implementations on PCs are vulnerable to physical side-channel attacks, and with what range, duration and techniques.

1.1 Our Contribution

In this paper, we preset the first physical side-channel attack on elliptic curve cryptography running on a PC. Moreover, our attack is *non-adaptive*, requiring decryption of a single, non-adaptively chosen ciphertext in order to extract the whole secret key by monitoring the target's electromagnetic (EM) field for just a few seconds.

We empirically demonstrate our technique on the ECDH public-key encryption algorithm used in OpenPGP [CDF$^+$07], as specified in RFC 6637 [Jiv12] and NIST SP800-56A [BCRS13] and as implemented in Libgcrypt 1.6.3 (which is the latest version at the time of writing this paper). To extract the secret key from the observed electromagnetic leakage, we utilize intricate time-frequency analysis techniques.

We demonstrate the attack's effectiveness by extracting keys from unmodified laptops running GnuPG, using their EM emanations as measured from an adjacent room through a wall (see Figure 6).

1.2 Attack Overview

The ECDH decryption consists primarily of multiplying the secret key (a scalar) by the curve point. The multiplication contains a sequence of point addition, doubling and inversion, and our approach utilizes the relation between the *operands* of these operations and the scalar. By asking for a decryption of a carefully-chosen ciphertext, we cause a specific curve point to appear as the operand in the elliptic curve additions. This point has a specific structure which causes an easy-to-observe effect on GnuPG's modular multiplication code. During the decryption of the chosen ciphertext, we measure the EM leakage of the target laptop, focusing on a narrow frequency band (frequencies in the range 1.5–2 MHz). After suitable signal processing, a clean trace is produced which reveals information about the operands used in the elliptic curve operations. This information, in turn, is used in order to reveal the secret key.

Note that our attacks *do not* assume any correlation between the sequence of elliptic curve double and add operations and the secret key. In particular, they work even if the scalar-by-point multiplication is implemented using only point additions.

1.3 Targeted Software and Hardware

Hardware. We target commodity laptop computers. During our experiments, we have tested numerous computes of various models and makes. The experiments described in this paper were conducted using a Lenovo 3000 N200 laptops, which exhibit a particularly clear signal. The attacks are completely non-intrusive: we did not modify the targets or open their chassis.

Software. We focus on Libgcrypt, which is popular cryptographic library that includes elliptic curve cryptography. Libgcrypt is used, in particular, by used in particular, by GnuPG 2.x [Gpga], which is very popular implementation of the OpenPGP standard [CDF$^+$07] used in applications such as encrypted mail and files. Concretely, we targeted Libgcrypt 1.6.3 (the latest versions at the time of writing), compiled using the MinGW GCC version 4.6.2 [Min].

Current Status. We worked with the developers of Libgcrypt and GnuPG to evaluate and deploy countermeasures preventing the attacks described in this paper (CVE 2015-7511). GnuPG's Libgcrypt 1.6.5, containing such countermeasures, was released simultaneously with the public posting of our results.

Chosen Ciphertext Injection. Our attack requires decryption of chosen ciphertexts. Conveniently, GnuPG and Libgcrypt are used by various applications, where they are used to decrypt externally-controlled inputs (the list of GnuPG frontends [Gpgb] contains dozens of such applications). One concrete attack vector was observed in [GST14], where Enigmail [Eni], a plugin for the Mozilla Thunderbird e-mail client, automatically decrypts incoming emails by passing them to GnuPG. Thus, it is possible to close the attack loop by remotely injecting the chosen ciphertext required by our attack into GnuPG via PGP/MIME-encoded e-mail [ETLR01]. Similar observations hold for the GnuPG Outlook plugin, GpgOL.

1.4 Related Work

For small devices, side-channel attacks have been extensively demonstrated, on numerous cryptographic implementations, using various channels, and in particular electromagnetic emanations starting with [AARR02, GMO01, QS01]. See [And08, KJJR11, MOP07] and the references therein.

Physical Attacks on ECC. Since the first (simulated) attacks by Coron [Cor99], there have been numerous physical side-channel attacks on implementations of Elliptic Curve Cryptography (ECC) on small devices; see the surveys [FGM$^+$10, FV12] and the references therein. However, such attacks typically target small devices and either utilize subtle physical effects which are only visible at bandwidths exceeding the device's clock rate, or attack naive implementations (such as the double-and-add algorithm). Three notable exceptions to the above approach are the attacks of Okeya and Sakura [OS02] and Walter [Wal04] attacking the Oswald-Aigner scalar randomization algorithm [OA01] assuming only the ability to distinguish between point addition and multiplication; the Refined Power Analysis attack of Goubin [Gou03]; and the Zero-Value Point Attacks of Akishita and Takagi [AT03].

Unfortunately, all of the above approaches have significant drawbacks in the case of GnuPG executed on PCs. Recording clock-rate scale signals (required for most attacks) from a full-fledged PCs computer running a GHz-scale CPU is difficult and requires expensive, cumbersome, and delicate lab equipment. The attacks of Okeya and Sakura [OS02] and Walter [Wal04] are only applicable to the Oswald-Aigner scalar randomization algorithm [OA01] (utilizing its non-determinism across various executions), which is not used by GnuPG. Finally, the attacks of Goubin [Gou03] and Akishita and Takagi [AT03] utilize adaptive chosen ciphertexts, requiring hundreds of ciphertexts in order to extract the secret scalar. Since in order to obtain a noise-free aggregate-trace several traces are required per ciphertext, overall the attacks of [Gou03] and [AT03] require the execution of several thousands of scalar-by-point multiplication operation, which is easily detectable.

Physical Side-Channel Attacks on PCs. Physical side-channel leakage from PCs have been demonstrated via voltage on USB ports [OS06] and power consumption [CMR$^+$13]. Cryptographically, physical side-channels were exploited for extracting keys from GnuPG's RSA and ElGamal implementations, using the acoustic channel [GST14], the chassis-potential channel [GPT14] and the electromagnetic channel [GPT14, GPPT15] (across several GnuPG versions, including both square-and-always-multiply and windowed exponentiation). On a related class of devices, namely smartphones, Goller and Sigl [GS15] showed electromagnetic attacks on square-and-sometimes-multiply RSA.

Software Side-Cache Attacks on GnuPG. Software-based side-channel key-extraction attacks on PCs were demonstrated using timing differences [BB05, BT11] and contention for various microarchitectural resources, such as caches [Ber05, Per05, OST06]. Recently such attacks were shown against GnuPG's implementation of RSA and ElGamal [YF14, YLG$^+$15], as well as elliptic-curve DSA [BvdPSY14, vdPSY15]. The latter attacks rely on the ability to distinguish between point doubling and point addition via cache access patterns, in order mount a lattice attack on DSA using partially known nonces. However, such types of attacks are not applicable for ECDH.

2 Cryptanalysis

2.1 GnuPG's Elliptic Curve Encryption Implementation

We attack OpenPGP's elliptic-curve public-key encryption scheme, called ECDH encryption, as specified in RFC 6637 [Jiv12] and defined as method `C(1e,1s,ECC CDH)` in NIST SP800-56A [BCRS13]. ECDH encryption is essentially Diffie-Hellman key exchange over a suitable elliptic curve, where one party's Diffie-Hellman message serves as that party's public key. The encryption operation runs the other party's part of the key exchange protocol against the public key, yielding a shared key. Decryption recomputes that shared key.

More explicitly, the ECDH encryption combines an elliptic-curve based Diffie-Hellman key exchange protocol and a symmetric-key cipher (typically AES), as follows. Given an elliptic curve group generator $\mathbb{G}$, key generation consists of generating a random scalar k. The secret key is then defined to be k while the public key is set to be $[k]\mathbb{G}$ (here and onward, we use additive group notation, and $[k]\mathbb{G}$ denotes scalar-by-point multiplication). Encryption of a message m is performed by generating a random scalar k', computing $[k']([k]\mathbb{G})$ and using the result in order to derive (using a key derivation function) a key x for the symmetric encryption algorithm. The message m is then symmetrically-encrypted using x, resulting in a ciphertext c'. The overall ciphertext is set to be $c = (c', [k']\mathbb{G})$. Decryption of a ciphertext $c = (c', [k']\mathbb{G})$ is done by computing $[k]([k']\mathbb{G})$, applying the key derivation function on the result to obtain a key x' for the symmetric encryption algorithm, and decrypting c' using x', resulting in a message m'. Since $[k]([k']\mathbb{G}) = [k']([k]\mathbb{G})$, we obtain that $x' = x$, resulting in $m' = m$.

Our attack deduces the secret key k from the side-channel leakage during the scalar-by-point multiplication $[k]\mathbb{G}'$ in the decryption.

GnuPG's Scalar-by-Point Multiplication. We now review GnuPG's implementation of the scalar-by-point multiplication operation which is used during the above-outlined ECDH encryption protocol. In order to perform the elliptic curve group operations as well as the large integer operations, GnuPG uses an internal mathematical library called MPI (based on GMP [Gmp]). For Weierstrass curves, GnuPG performs the scalar-by-point multiplication operation using the standard double-and-add algorithm (Algorithm 1), maintaining the scalar in *non-adjacent form (NAF)* which we now discuss. **Non-Adjacent Form Representation.** Introduced by Reitwiesner [Rei60], the non-adjacent form is a common generalization of the standard binary representation of integers, allowing for both positive and negative bits. For example, the 4-digit NAF representation of 7 is (1,0,0,-1) compared to its binary representation (0,1,1,1). The main advantage of using a NAF representation is that it minimizes the number of non-zero digits from about 1/2 for the binary representation to about 1/3. Since every non-zero digit requires a point addition operation, using a NAF representation minimizes the number of point additions. Thus, most modern representations of elliptic curve cryptography typically represent scalars in using NAF.

We proceed to describe GnuPG's point addition operation, used in lines 6 and 9. Later in Section 2.2 we will show how to exploit GnuPG's implementation of point addition in order to achieve key extraction.

GnuPG's Point Addition. GnuPG stores elliptic curve points using projective coordinates. Each point is a tuple (x, y, z) where each element is a large integer stored using GnuPG's mathematical library, MPI. Large integers are stored by MPI as arrays of *limbs*, which are 32-bit words (on the x86 architecture used in our tests). Algorithm 2 is a pseudocode of GnuPG's point addition operation. Notice the multiplication by y_2 in line 10. We will now show how this multiplication can be exploited in order to distinguish between -1 and 1 valued NAF digits of k, resulting in a complete key extraction.

Algorithm 1 GnuPG's scalar-by-point multiplication operation (simplified).

Input: A positive scalar k and an elliptic-curve point $\mathbb{P}$, where $k_{n-1} \cdots k_0$ is the NAF representation of k, that is $k = \sum_{i=0}^{n-1} 2^i \cdot k_i$ and $k_i \in \{-1,0,1\}$ for all $i = 0, \cdots, n-1$.

Output: $[k]\mathbb{P}$.

```
1:  procedure POINT_MUL($k, \mathbb{P}$)
2:      $\mathbb{A} \leftarrow \mathbb{P}$
3:      for $i \leftarrow n-1$ to $0$ do
4:          $\mathbb{A} \leftarrow [2]\mathbb{A}$
5:          if $k_i = 1$ then
6:              $\mathbb{A} \leftarrow \mathbb{A} + \mathbb{P}$
7:          if $k_i = -1$ then
8:              $\mathbb{P}' \leftarrow [-1]\mathbb{P}$
9:              $\mathbb{A} \leftarrow \mathbb{A} + \mathbb{P}'$
10:     return $\mathbb{A}$
```

Algorithm 2 GnuPG's point addition operation (simplified).

Input: Two points $\mathbb{P}_1 = (x_1, y_1, z_1)$ and $\mathbb{P}_2 = (x_2, y_2, z_2)$ in projective coordinates on an elliptic-curve based group of order p.

Output: A point $P_3 = (x_3, y_3, z_3)$ in projective coordinates such that $\mathbb{P}_3 = \mathbb{P}_2 + \mathbb{P}_1$.

```
procedure POINT_ADD($\mathbb{P}_1, \mathbb{P}_2$)
    if $z_1 = 0$ then
        return $\mathbb{P}_2$                                   ▷ $\mathbb{P}_1$ is at infinity
    if $z_2 = 0$ then
        return $\mathbb{P}_1$                                   ▷ $\mathbb{P}_2$ is at infinity
    $l_1 \leftarrow x_1 z_2^2 \bmod p$
    $l_2 \leftarrow x_2 z_1^2 \bmod p$
    $l_3 \leftarrow l_1 - l_2 \bmod p$
    $l_4 \leftarrow y_1 z_2^3 \bmod p$
    $l_5 \leftarrow y_2 z_1^3 \bmod p$
    $l_6 \leftarrow l_4 - l_5 \bmod p$
    if $l_3 = 0$ and $l_6 = 0$ then
        return (1,1,0)          ▷ $\mathbb{P}_1$ is the inverse of $\mathbb{P}_2$ thus the result is infinity
    $l_7 \leftarrow l_1 + l_2 \bmod p$
    $l_8 \leftarrow l_4 + l_5 \bmod p$
    $z_3 \leftarrow z_1 z_2 l_3 \bmod p$
    $x_3 \leftarrow l_6^2 - l_7 l_3^2 \bmod p$
    $l_9 \leftarrow l_7 l_3^2 - 2x_3 \bmod p$
    $y_3 \leftarrow (l_9 l_6 - l_8 l_3^3)/2 \bmod p$
    return $(x_3, y_3, z_3)$
```

2.2 ECDH Attack Algorithm

Let *DA-sequence* denote the sequence of double and add operations performed in lines 4, 6 and 9 of Algorithm 1. Notice that it is possible to deduce all the locations of zero valued NAF digits of k by simply observing the DA sequence performed by Algorithm 1. However, since k is given in a NAF representation, recovering the DA-sequence alone is not enough for achieving key extraction: there remains an ambuguity between -1 and 1 valued NAF digits of k, since point addition is executed in both cases (in addition to point doubling).

Observing Point Inversions. An immediate approach for distinguishing between 1 and -1 valued NAF digits would consist of attempting to observe the point inversion operation performed in line 8. However, for Weierstrass curves, inverting a point $\mathbb{P} = (x, y)$ on an elliptic-curve group of order p, simply requires computing the inverse of y modulo p. This operation is too fast for us to observe in our low-bandwidth setting. Moreover, fact that point inversion is performed at every -1-valued digit of the NAF form of k constitutes a side-channel weakness in GnuPG's point multiplication code, which is unlikely to be present in a more robust implementation. We thus do not utilize this observation for our attack.

We proceed to describe how, by using a chosen ciphertext, an attacker can distinguish between the add operations performed by line 6 and the add operations performed by line 9. This information, together with the DA-sequence is enough to recover the secret scalar k.

Distinguishing Between the NAF Digits of $\boldsymbol{k}$. Let $\mathbb{Q} = (x, y)$ be a point with small y (containing few limbs) and a random-looking (full-sized) x. Consider performing an ECDH encryption operation of a ciphertext $(c', \mathbb{Q})$ for some c'. Since GnuPG's internal representation uses projective coordinates, the point $\mathbb{Q}$ converted to a $\mathbb{P}$ in a projective representation $\mathbb{P} = (x, y, 1)$ and it is then passed to Algorithm 1. Next, $\mathbb{P}$ is used in lines 6 and 9 thereby affecting the leakage produced by each iteration of the main loop of Algorithm 1 as follows.

1. $\boldsymbol{k_i = 0}$. In this case only a point doubling operation is performed by Algorithm 1. Thus, as mentioned before, these digits are immediately recoverable from the DA-sequence since any double operation which is not followed by an add operation corresponds to a zero valued digit of k.

2. $\boldsymbol{k_i = 1}$. In this case $\mathbb{P}$ is passed as is to the point addition routine (Algorithm 2) as its second argument $\mathbb{P}_2$. Since y is small, the first operand, y_2, of the multiplication in line 10 is only a few limbs long.

3. $\boldsymbol{k_i = -1}$. In this case the point $\mathbb{P}$ is first inverted by line 8. For Weierstrass curves, point inversion corresponds to computing the modular inverses of the y coordinate, so the y coordinate of $\mathbb{P}'$ is random looking. This $\mathbb{P}'$ is passed to the point addition routine (Algorithm 2) as its second argument $\mathbb{P}_2$. This makes the first operand, y_2, of the multiplication in line 10 be random looking and (likely) full length.

By observing the side-channel leakage produced by Algorithm 1, we will be able to recover the DA-sequence, and also distinguish, in each invocation the multiplication in line 10 of Algorithm 2, whether the first operand is short or full length. As explained above, this information is enough in order to recover the secret scalar k.

2.3 Attacking the Always-Add Algorithm

In this section we generalize the above method for attacking a variant of Algorithm 1 where the point doubling operation is implemented using point addition. That is, we assume that line 4 is replaced by $\mathbb{A} \leftarrow \mathbb{A} + \mathbb{A}$. In particular, in this section, we do not assume that it is possible to immediately distinguish the point additions performed by lines 6 and 9 from the point doublings performed by line 4. As we show, it is possible to utilize two chosen ciphertexts in order to recover the DA-sequence as well as, for every addition operation, whether the corresponding NAF digit is 1 or -1.

Revealing the 1 Digits of $\boldsymbol{k}$. As in Section 2.2, the attacker requests a decryption of a point $\mathbb{P}$ with small y coordinate. As discussed in Section 2.2, this creates a distinguishable leakage every time that $k_i = 1$ during the execution of the main loop of Algorithm 1, thereby revealing the locations in the DA-sequence of all such digits. **Revealing the $\boldsymbol{-1}$ Digits of $\boldsymbol{k}$.** Next, the attacker selects a point $\mathbb{P}$ whose *inverse* has a small y coordinate, and requests an ECDH decryption of $(c, \mathbb{P})$ for some arbitrary value c. During the main loop of Algorithm 1 every time that $k_i = -1$ the inversion of $\mathbb{P}$, denoted by $\mathbb{P}'$, is passed to the point addition routine. Since $\mathbb{P}$ was chosen such that $\mathbb{P}'$ has a small y coordinate, as discussed in Section 2.2 this creates a distinguishable leakage every time that $k_i = -1$ during the main loop of Algorithm 1, thereby revealing the locations in the DA-sequence of all such digits. **Key Extraction.** At this point the attacker has recovered the locations in the DA-sequence of all point additions as performed by lines 6 and 9 of Algorithm 1. Moreover, for each point addition, the attacker has recovered the corresponding value of k_i. Thus, all remaining operations in the DA-sequence are in fact points doublings. Using this information at hand, the scalar k can be recovered.

3 Signal Analysis and Experimental Results

3.1 Experimental Setup

This section describes the lab setup used for characterizing the EM leakage from target computers at frequencies of 0–5 MHz. We have also constructed a more realistic setup, described in see Section 3.3.

Probe. To measure the EM leakage from the target laptop with high spatial precision, we used a Langer LF-R 400 near field probe (a 25mm loop probe, 0–50 MHz). The location of the probe relative to the laptop body greatly affects the measured signal. In our experiments, the best signal quality was obtained close to the CPU's voltage regulator, which on most laptops is located in the rear left corner. We thus placed the probe at that position, without any chassis intrusion or other modification to the target laptop.

Amplification and Digitization. To amplify the signal measured by the probe we used a (customized) Mini-Circuits ZPUL-30P amplifier, providing 40 dB of gain. The output of the amplifier was then low-pass filtered at 5 MHz and digitized using a National-Instruments PCI 6115 data acquisition device sampling at 10 Msample/sec with 12 bits of ADC resolution.

3.2 Signal Analysis

Scalar-Dependant Leakage. As an initial confirmation of the existence of scalar-dependent leakage from the point multiplication, Figure 1 shows five distinct leakage patterns, obtained by multiple invocation (in sequence) of Algorithm 1 using the same point $\mathbb{P}$ with small y coordinate and five different values of the scalar k. Such key-dependent leakage was observed on many target laptops, often in multiple frequency bands.

Observing Figure 1, notice that for periodic scalars the spectral signature of the leakage signal has strong side-bands surrounding a central carrier frequency. This is a strong indication of a key-dependent modulation signal on a carrier frequency (analogously to modulations observed in [GPT14, GPPT15])

Demodulation. We proceed to describe our signal processing methodology demodulating the acquired signal and deducing the DA-sequence, as well as for distinguishing between -1 and 1 NAF digits, for complete key extraction.

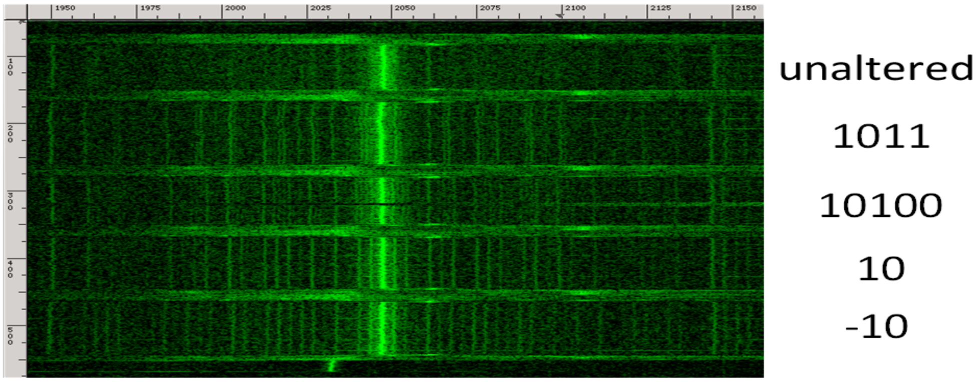
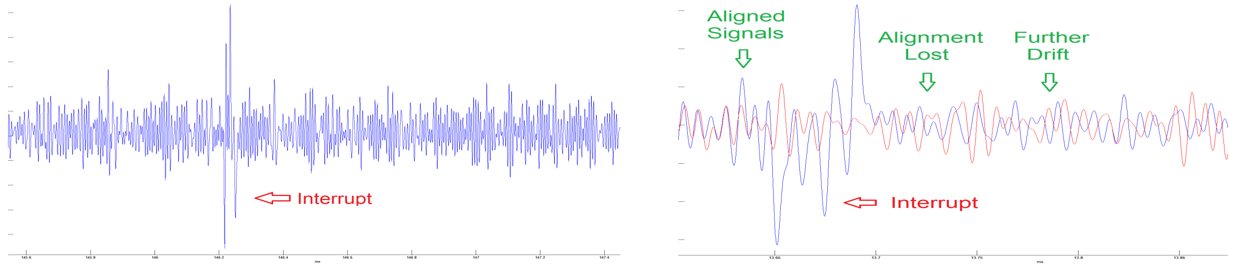


Figure 1: EM measurement (0.5 sec, 1.95-2.15 MHz) of five scalar-by-point multiplication operations using the NISTP-521 curve executed on a Lenovo 3000 N200 laptop. The scalar was overridden to be the 521-digit number obtained by repeating the pattern written to the right. In all cases, the curve point had the same random-looking x coordinate and a small y coordinate.



(a) Part of a trace obtained during a single decryption (after FM demodulation and filtering). Note the interrupt corrupting part of the signal.

(b) Two demodulated traces obtained during two decryption operations, using the same ciphertext and key. Note the loss of alignment due to the interrupt.

Figure 2: Frequency demodulated traces obtained from a single decryption operation.

For each target, we manually scanned the spectrum and chose the carrier frequency exhibiting the clearest modulation side-bands. After analog filtering and sampling, we used a digital band pass filter to suppress all frequencies outside the band of interest. As in the case of [GPT14, GPPT15], the key-dependent signal turned out to be frequency modulated (FM) on the carrier signal. Demodulation was performed using the digital Hilbert transform, followed by further filtering. Figure 2(a). shows an example of the resulting trace.

Obtaining a Clear Trace. Similarly to [GPT14, GPPT15], parts of each demodulated decryption trace were occasionally corrupted by strong disturbances, e.g., due to timer interrupts in the target laptop. But even ignoring these, a simple visual inspection of the trace in Figure 2(a) reveals no immediately obvious patterns or clues about the scalar k or the inner workings of Algorithm 1. In order to obtain a clearer trace and remove the interrupts, we used a multi-step procedure involving the aggregation of several dozen recorded decryption traces, as follows.

Interrupts and Drifts. To aggregate traces, we first attempted simple alignment via correlation. Unfortunately, the traces exhibited slow random drifts relative to each other, so that full alignment of entire traces proved difficult. In addition, interrupts induced further random delays in each trace relative to other traces, as well as signal distortion. See Figure 2(b).

Initial Alignment. Despite the relative distortion between decryption traces, we did notice that

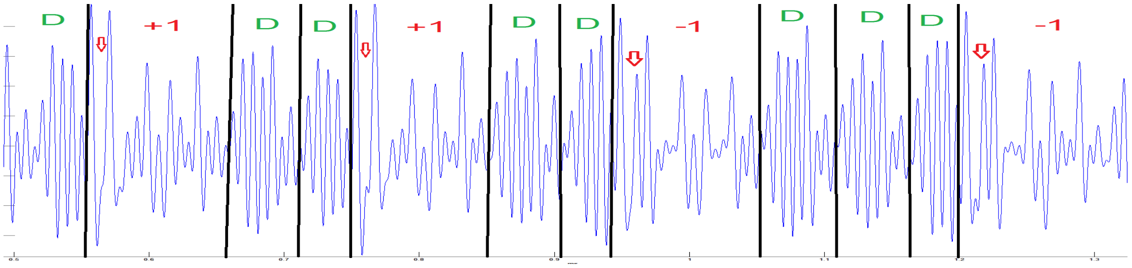


Figure 3: Part of an aggregated trace obtained from several decryption operations during our attack. The double operations is marked with D and the add operations are marked with the corresponding bit of k_i (either 1 or -1). The red arrows mark the differences between additions performed by lines 6 and 9 of Algorithm 1. Notice that the difference occurs at the begining of each addition operation, as expected from Algorithm 2.

a short trace segment immediately preceding each decryption operation was relatively similar across all measurements, rarely having any interrupts or drifts. We thus used this common segment to perform an initial alignment of all decryption traces, using simple correlation, as follows. We first, chose a reference trace at random and aligned the initial segment of all other traces relative to it. If the initial segment of the reference trace was corrupted due to noise or distortion, the current reference trace was discarded and a new one chosen. If the initial segment of one of the other traces did not align well with that of the reference trace, it was also discarded.

Gradual Alignment Correction. After achieving initial alignment of all decryption traces, we compensated for the gradual drifts of the traces relative to the reference trace by performing individual alignment correction as follows. Each trace was independently compared with the reference trace, by simultaneously inspecting it from beginning to end. Periodically, the relative phase lag between the two traces was estimated by cross-correlating a short local section in both traces. Any detected misalignment was immediately corrected. If an interrupt was detected in one of the traces during this process, the delay it induced was also corrected. Interrupts are easily detectable since they cause large frequency fluctuations. The above process was performed independently for each trace, always in respect to the original reference trace.

Trace Aggregation. Even after the alignment correction process described above, direct aggregation of the traces did not produce an aggregated trace with sufficient fidelity. In order to fine-tune the alignment and facilitate the aggregation process, we broke each trace down into a large number of shorter segments, each corresponding to roughly 20 double and add operations. These were in turn aligned again across all traces via correlation with a corresponding randomly-chosen reference segment. After this final alignment step, segments were aggregated across all traces via a mean-median filter hybrid. For each segment and at each time point, the samples across all traces were sorted, and several lowest and highest values discarded. The rest of the samples were consequently averaged, resulting in distortion-free aggregate trace segments. Figure 3 shows an example of such an aggregate segment. The individual double and add operations can now clearly be seen.

Key extraction. For key extraction, we must deduce from each aggregated segment the partial DA-sequence it contains, as performed by Algorithm 1. Moreover, for each addition operation in the partial DA-sequence, we must also somehow distinguish whether the corresponding NAF digit is 1 or -1. Obtaining this information will result in several dozen sequences of trinary bits each representing a fragment of the NAF representation of the secret constant k. To facilitate the reconstruction of k from its fragments, we chose to take the aggregate trace segments mentioned in the previous section to be largely overlapping. In such a case, consecutive fragments of the NAF

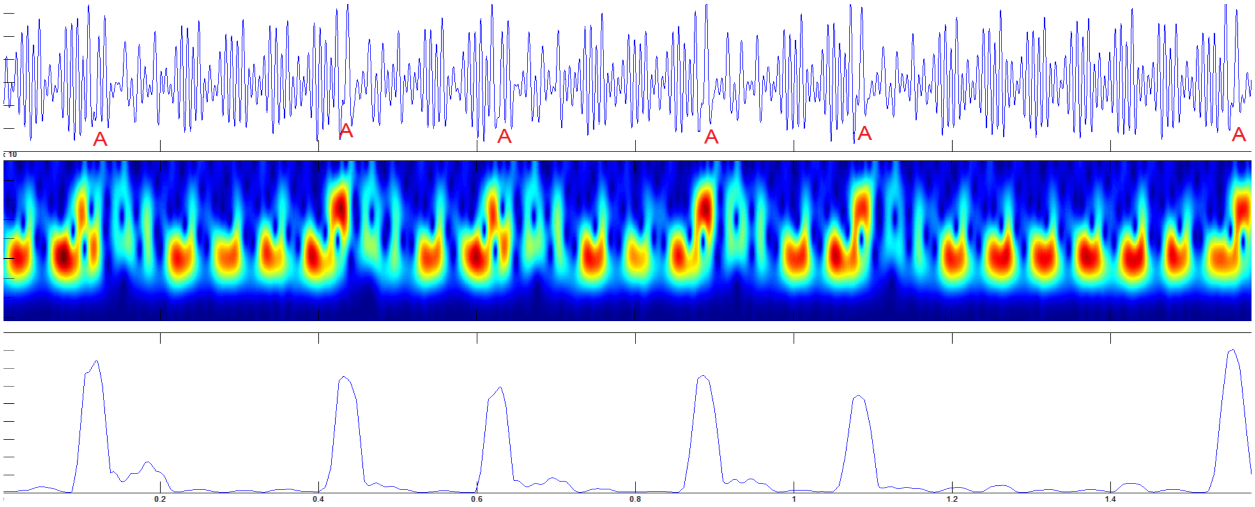


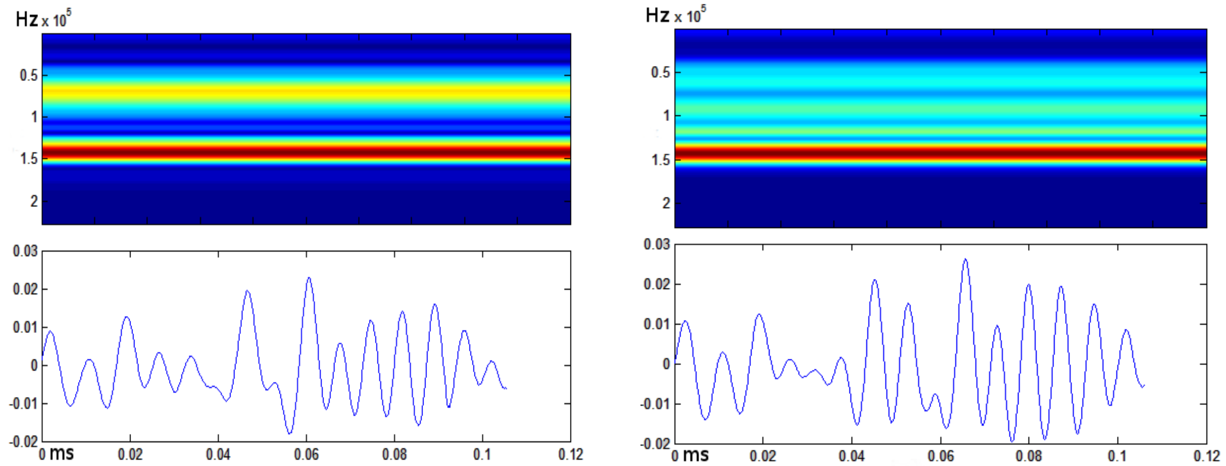
Figure 4: Several stages of our approach for distinguishing between double and add operations. The topmost figure is the aggregated segment corresponding to the bottom two figures, with the locations of addition operations marked. The middle figure is the spectrogram of the aggregated segment with blue denotes frequencies with low-energy while red denotes frequencies with high energy. In this figure the horizontal frequency is time (0-1.6 msec) while the vertical axis is frequency (0–400 kHz). The bottom figure represents the final result of our approach clearly showing the locations of the addition operations, obtained by performing the procedure described above.

representation of k will have many overlapping bits, allowing for a unique reconstruction.

We now describe the process of extracting the partial DA-sequence from each aggregated segment as well as the process of determining whether the corresponding NAF digit is 1 or -1.

Extracting the Partial DA-Sequence. Although the sequence of double and add operations can be identified in Figure 3 by careful observation, it is not clear how it can be extracted automatically and reliably. Attempting this in the (post-FM-demodulation) time domain appears difficult since both double and add operations are comprised of largely similar peaks. Instead, we utilize an alternative approach, utilizing the information present in the (post-FM-demodulation) frequency domain. The top and middle parts of Figure 4 show an aggregated segment along with its corresponding spectrogram. It can be seen that the addition and doubling operations generate energy in two mostly separate frequency bands. We consequently focus on the upper band which contains most of the energy of addition operations, and filter each aggregated segment around this frequency-band. Notice that each doubling operation also contributes some small amount of energy to this band, which may create false positives. In order to reliably extract the timings of all addition operations, we multiply the energy in the upper band with its own derivative with respect to time. In this manner we are able to enhance energy peaks that are already both strong and sharply-rising, and attenuate any other peaks. After additional smoothing and equalization, we obtain the trace in the bottom part of Figure 4 in which the occurrences of addition operations are clearly visible. The timings of doubling operations are then inferred by the time-lapse between additions, thus recovering the partial DA-sequence present in each aggregated segment.

Distinguishing Between 1 and -1. While the spectrogram in Figure 4 proved very useful in identifying sequences of double and add operations, it is far less effective in determining whether the NAF digit corresponding to an add operation is 1 or -1. The leakage induced by our chosen ci-



(a) An aggregated segment of an addition operations corresponding to a 1 NAF digit

(b) An aggregated segment of an addition operations corresponding to a -1 NAF digit

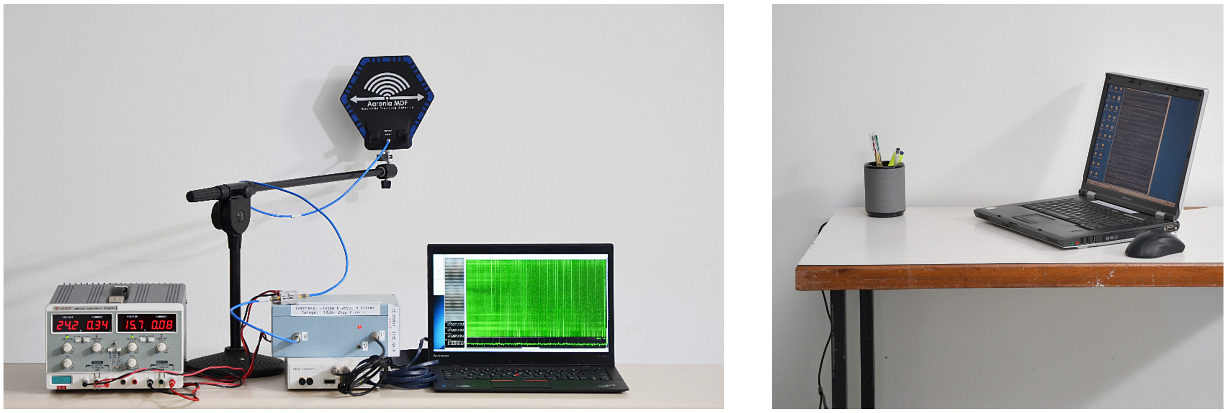
Figure 5: Zoomed-in views (bottom) and spectrograms (top) of add operations corresponding to 1 and -1 NAF digits. Note the energy difference in the 50–125 kHz band between the two signals. This difference is consistent across all add operations, and can be used to differentiate between them.

phertext is slight and only affects one of several modular multiplications performed by Algorithm 2. Since the leakage is so short lived, it is difficult to differentiate between the frequency signatures of the two cases. In order to overcome the issue we use the exact timings of the add operations (which are already known from the previous step). For each add operation we zoom in on each addition operation in the original aggregated trace using the timings obtained from the previous step. In this manner we discard anything unrelated to the addition operation itself. We then plot a spectrogram using a large time window, thereby increasing the frequency resolution at the price of time resolution. This reveals consistent differences between addition operations corresponding to 1 and -1 NAF digits, in two frequency bands, see Figure 5. This difference allows us to consistently differentiate between the two add operations (corresponding to 1 and -1 NAF digits), resulting in a reliable key extraction.

Overall Attack Performance. Applying our attack to a randomly generated ECDH NISTP-521 key, by measuring the EM emanations of a Lenovo 3000 N200 target, we have extracted the secret scalar except its first 5 NAF digits, with an error of two digits. During the attack we have used traced obtained form 75 decryption operations, each lasting about 0.05 sec, yielding a total measurement time of about $75 \cdot 0.05 = 3.75$ sec.

3.3 Measuring the EM Leakage Through a Wall

In order to eavesdrop on the EM leakage emanating from target computers in surrounding rooms, we have constructed a more portable experimental setup which we now discuss. **Antenna.** We have used an Aaronia Magnetic Direction Finder MDF 9400 antenna, designed for 9 kHz–400 MHz. This is essentially a tuned loop antenna. **Amplification and Digitization.** The signals produced by the antenna were amplified first by a Mini-Circuits ZFL-1000 amplifier and then by a (customized) Mini-Circuits ZPUL-30P amplifier, providing a total of gain of approximately 60 dB (at the frequency of interest). The resulting signal was then low-pass filtered at 5 MHz and digi-



(a) Attacker's setup for capturing EM emanations. Left to right: power supply, antenna on a stand, amplifiers, software defined radio (white box), analysis computer.

(b) Target (Lenovo 3000 N200), performing ECDH decryption operations, on the other side of the wall.

Figure 6: Attacking a target computer in an adjacent room, across a wall.

tized using an Ettus Research USRP N200 software defined radio, equipped with a LFRX daughter board, at 10 Msample/sec. **Target Placement.** For this experiment, the target laptop was placed in a room adjacent to the attacker's experimental setup, separated by a standard drywall (15 cm thick, reinforced with metal studs). The location and orientation of the antenna greatly affects the resulting signal. In our experiments, we have placed the antenna on the opposite side of the wall from the target computer's voltage regulator, with the antenna's loop plane parallel to the wall surface. See Figure 6.

Overall Attack Performance. Applying our attack and signal processing techniques to a target laptop (Lenovo 3000 N200) located in the adjacent room, we have successfully extracted the secret scalar of a randomly generated ECDH NISTP-521 key except its first 5 NAF digits and with an error of two digits. For the attack we have used traces collected by measuring the target's EM leakage during 66 decryption operations, each lasting about 0.05 sec. This yields a total measurement time of about 3.3 sec.

4 Conclusion

This paper demonstrates the first side-channel attack on PC implementations of elliptic curve cryptology. Our techniques do not assume the leakage of secret-key material via the sequence of elliptic curve double and add operations. Instead our attacks rely on a strong correlation between the *operands* of elliptic curve addition operation and the secret key. By injecting carefully chosen ciphertexts, we make the operands to GnuPG modular multiplication routine highly distinguishable, even by low-bandwidth measurements. Since the operands of the elliptic curve addition operation are highly correlated with the secret key, we are able to completely recover the key within only a few seconds of measurements.

Software Countermeasures. Our attacks extract the secret key by observing the leakage created during the decryption of a carefully chosen ciphertext (curve points) which creates some mathematical structure in the operands of the elliptic curve addition operation. We now review the common set of countermeasures for preventing such chosen ciphertext attacks, see [FGM$^+$10, FV12] for extended discussions.

Scalar Randomization and Splitting. Many side-channel attacks relay on averaging the leakage during several decryption operations on order to achieve key extraction. A scalar randomization countermeasure [Cor99] prevents such averaging by adding to the scalar a random multiple of the group order before performing the scalar-by-point multiplication operation. This changes the sequence of elliptic curve double and add operations performed during different decryption operations, thus hindering the averaging operation. Another common and similar countermeasure splits the secret scalar k in into n parts $k_1, \cdots, k_n$ such that $k = \sum_{i=1}^{n} k_i$, performs the scalar-by-point multiplication operation separately on each k_i and them combines the result [CJ03].

While such a countermeasure is indeed effective against our attack (since it requires traces obtained from several decryption operations), it will not stop chosen ciphertext attacks that only rely on a single trace for key extraction.

Point Blinding. This method protects the scalar k multiplied with a ciphertext point $\mathbb{P}$, by first generating a random point $\mathbb{R}$, computing $k(\mathbb{P}+\mathbb{R})$ and then subtracting $k\mathbb{R}$ from the result [Cor99]. Such a countermeasure will completely block chosen ciphertext attacks since the attacker is no longer able to carefully chose a point $\mathbb{P}$ to be multiplied with k. However, the effect on performance of this countermeasure is often significant, since now two scalar-by-point multiplication operations have to be performed per decryption.

Future Work. While in the past few years there have been several physical key-extraction attacks on full fledged-PC computers [GST14, GPT14, GPPT15], all of these attacks relied on a carefully chosen ciphertext and targeted various public key encryption schemes. We pose, as intriguing open problems, the challenges of non-chosen ciphertext attacks as well as attacking other cryptographic primitives (such as symmetric encryption). Finally, our attacks utilized traces obtained from about 70 decryption operations in order to extract the secret key. We pose the task of minimizing this number as another open problem.

Acknowledgments

We thank Werner Koch, lead developer of GnuPG, for the prompt response to our disclosure and the productive collaboration in adding suitable countermeasures.

This work was sponsored by the Check Point Institute for Information Security; by the European Union's Tenth Framework Programme (FP10/2010-2016) under grant agreement no. 259426 ERC-CaC, by a Google Faculty Research Award, by the Israeli Ministry of Science and Technology; by the Israeli Centers of Research Excellence I-CORE program (center 4/11); by the Leona M. & Harry B. Helmsley Charitable Trust; 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 *Workshop on Cryptographic Hardware and Embedded Systems (CHES) 2002*, pages 29–45. Springer, 2002.

[And08] Ross J. Anderson. *Security Engineering — A Guide to Building Dependable Distributed Systems (2nd ed.).* Wiley, 2008.

[AT03] Toru Akishita and Tsuyoshi Takagi. Zero-value point attacks on elliptic curve cryptosystem. In *International Conference on Information Security (ISC) 2003*, pages 218–233, 2003.

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

[BCRS13] Elaine Barker, Lily Chen, Allen Roginsky, and Miles Smid. NIST SP 800-56a: Recommendation for pair-wise key establishment schemes using discrete logarithm cryptography (revision 2), 2013.

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

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

[BvdPSY14] Naomi Benger, Joop van de Pol, Nigel P. Smart, and Yuval Yarom. "ooh aah... just a little bit" : A small amount of side channel can go a long way. In *Cryptographic Hardware and Embedded Systems (CHES) 2014*, pages 75–92, 2014.

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

[CJ03] Mathieu Ciet and Marc Joye. (virtually) free randomization techniques for elliptic curve cryptography. In *International Conference Information and Communications Security (ICICS) 2003*, pages 348–359. Springer, 2003.

[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 2013*, pages 700–717. Springer, 2013.

[Cor99] Jean-Sébastien Coron. Resistance against differential power analysis for elliptic curve cryptosystems. In *Cryptographic Hardware and Embedded Systems (CHES) 2002*, pages 292–302, 1999.

[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, 2001. URL: `http://www.ietf.org/rfc/rfc3156.txt`.

[FGM$^+$10] Junfeng Fan, Xu Guo, Elke De Mulder, Patrick Schaumont, Bart Preneel, and Ingrid Verbauwhede. State-of-the-art of secure ECC implementations: A survey on known side-channel attacks and countermeasures. In *Proceedings of the 2010 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST) 2010*, pages 76–87, 2010.

[FV12] Junfeng Fan and Ingrid Verbauwhede. An updated survey on secure ECC implementations: Attacks, countermeasures and cost. In *Cryptography and Security: From Theory to Applications - Essays Dedicated to Jean-Jacques Quisquater on the Occasion of His 65th Birthday*, pages 265–282, 2012.

[GMO01] Karine Gandolfi, Christophe Mourtel, and Francis Olivier. Electromagnetic analysis: concrete results. In *Workshop on Cryptographic Hardware and Embedded Systems (CHES) 2001*, pages 251–261. Springer, 2001.

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

[Gou03] Louis Goubin. A refined power-analysis attack on elliptic curve cryptosystems. In *International Workshop on Theory and Practice in Public Key Cryptography (PKC) 2003*, pages 199–210, 2003.

[Gpga] GNU Privacy Guard. URL: https://www.gnupg.org.

[Gpgb] GnuPG Frontends. URL: https://www.gnupg.org/related_software/frontends.html.

[GPPT15] Daniel Genkin, Lev Pachmanov, Itamar Pipman, and Eran Tromer. Stealing keys from PCs using a radio: Cheap electromagnetic attacks on windowed exponentiation. In *Workshop on Cryptographic Hardware and Embedded Systems (CHES) 2015*, pages 207–228, 2015. Extended version: Cryptology ePrint Archive, Report 2015/170.

[GPT14] Daniel Genkin, Itamar Pipman, and Eran Tromer. Get your hands off my laptop: Physical side-channel key-extraction attacks on PCs. In *Workshop on Cryptographic Hardware and Embedded Systems (CHES) 2014*, pages 242–260. Springer, 2014. Extended version: Cryptology ePrint Archive, Report 2014/626.

[GS15] Gabriel Goller and Georg Sigl. Side channel attacks on smartphones and embedded devices using standard radio equipment. In *International Workshop on Constructive Side-Channel Analysis and Secure Design (COSADE) 2015*, pages 255–270. Springer, 2015.

[GST14] Daniel Genkin, Adi Shamir, and Eran Tromer. RSA key extraction via low-bandwidth acoustic cryptanalysis. In *CRYPTO 2014*, pages 444–461 (vol. 1). Springer, 2014. Extended version: Cryptology ePrint Archive, Report 2013/857.

[Jiv12] A. Jivsov. Elliptic curve cryptography (ECC) in OpenPGP. RFC 4880, 2012.

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

[Min] Minimalist GNU for Windows. URL: http://www.mingw.org.

[MOP07] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power Analysis Attacks — Revealing the Secrets of Smart Cards.* Springer, 2007.

[OA01] Elisabeth Oswald and Manfred Josef Aigner. Randomized addition-subtraction chains as a countermeasure against power attacks. In *Cryptographic Hardware and Embedded Systems (CHES) 2001*, pages 39–50, 2001.

[OS02] Katsuyuki Okeya and Kouichi Sakurai. On insecurity of the side channel attack countermeasure using addition-subtraction chains under distinguishability between addition and doubling. In *Australian Conference on Information Security and Privacy (ACISP) 2002*, pages 420–435, 2002.

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

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

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

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

[Rei60] George W. Reitwiesner. Binary arithmetic. *Advances in Computers*, 1:231–308, 1960.

[vdPSY15] Joop van de Pol, Nigel P. Smart, and Yuval Yarom. Just a little bit more. In *RSA Conference Cryptographers' Track (CT-RSA) 2015*, pages 3–21, 2015.

[Wal04] Colin D. Walter. Issues of security with the oswald-aigner exponentiation algorithm. In *RSA Conference Cryptographers' Track (CT-RSA) 2004*, pages 208–221, 2004.

[YF14] Yuval Yarom and Katrina Falkner. FLUSH+RELOAD: a high resolution, low noise, L3 cache side-channel attack. In *USENIX Security Symposium 2014*, pages 719–732. USENIX Association, 2014.

[YLG$^+$15] Yuval Yarom, Fangfei Liu, Qian Ge, Gernot Heiser, and Ruby B. Lee. Last-level cache side-channel attacks are practical. In *IEEE Symposium on Security and Privacy (S&P) 2015*. IEEE, 2015.