J Cryptogr Eng (2011) 1:5–27
DOI 10.1007/s13389-011-0006-y

REGULAR PAPER

Introduction to differential power analysis

Paul Kocher · Joshua Jaffe · Benjamin Jun · Pankaj Rohatgi

Received: 19 November 2010 / Accepted: 6 January 2011 / Published online: 3 March 2011

Abstract The power consumed by a circuit varies according to the activity of its individual transistors and other components. As a result, measurements of the power used by actual computers or microchips contain information about the operations being performed and the data being processed. Cryptographic designs have traditionally assumed that secrets are manipulated in environments that expose no information beyond the specified inputs and outputs. This paper examines how information leaked through power consumption and other side channels can be analyzed to extract secret keys from a wide range of devices. The attacks are practical, non-invasive, and highly effective—even against complex and noisy systems where cryptographic computations account for only a small fraction of the overall power consumption. We also introduce approaches for preventing DPA attacks and for building cryptosystems that remain secure even when implemented in hardware that leaks.

Keywords Differential power analysis · DPA · SPA · Side-channel attacks · Tamper resistance · Cryptanalysis

P. Kocher · J. Jaffe (✉) · B. Jun · P. Rohatgi
Cryptography Research, Inc, 575 Market Street, 11th Floor, San Francisco, CA 94105, USA
e-mail: josh@cryptography.com
URL: http://www.cryptography.com

P. Kocher
e-mail: paul@cryptography.com

B. Jun
e-mail: ben@cryptography.com

P. Rohatgi
e-mail: pankaj@cryptography.com

1 Background

Because attacks that involve multiple layers of a system are difficult to predict and model, security vulnerabilities often result from unanticipated interactions between components and layers. If algorithm designers, software developers, and hardware engineers do not collaborate and understand each other's work, security assumptions made in one layer of a system may not match the actual properties of other layers.

Many techniques have been designed for testing cryptographic algorithms in isolation. For example, differential cryptanalysis [1] and linear cryptanalysis [2] can exploit extremely small statistical characteristics in a cipher's inputs and outputs. Modern ciphers are designed to resist such attacks. Such analysis only applies, however, to one part of a system's architecture—an algorithm's mathematical structure.

Resistance to cryptanalysis is not sufficient to create secure cryptosystems in practice. Even a correct implementation with strong algorithms and protocols is not necessarily secure, since vulnerabilities can arise from other layers of the implementation. For example, security can be compromised by defective computations [3,4]. Attacks using timing information [5,6] as well as data collected using invasive measuring techniques [7,8] have also been demonstrated. The US government has invested considerable resources in the classified TEMPEST program [9] to prevent sensitive information from leaking through electromagnetic emanations.

In this paper, we introduce differential powerful analysis (DPA), simple power analysis (SPA), as well as several related techniques. These attacks leverage measurements of a target device's power consumption (or other side channels) to extract secret keys. The methods are effective against implementations of all major algorithms.

In the years since we first documented DPA, a tremendous amount of research has been published on the subject [10]. The attacks have been implemented against hundreds of devices, including implementations in ASICs, FPGAs, and software. The targets range from tiny single-purpose chips to complex devices whose power measurements are noisy and obfuscated by unpredictable parallel operations. In short, the evolution of power analysis attacks conforms to the adage attributed to the National Security Agency: "Attacks always get better; they never get worse."

1.1 Organization of this paper

Section 2 introduces power analysis attacks, beginning in Sect. 2.1 with an overview of power traces and their properties. Section 2.2 then presents a straightforward DPA attack.[1] Section 3 then discusses simple power analysis and related methods. Section 4 then returns to the topic of DPA, and explores both the statistical properties of the attack as well as practical techniques to make the attack process more efficient. Variants of DPA are described in Sect. 5. Section 6 summarizes the types of countermeasures that can be used to defend against side-channel attacks. Finally, we conclude in Sect. 7.

2 Introduction to power analysis

Most modern cryptographic devices are implemented using semiconductor logic gates, which are constructed out of transistors. Electrons flow across the silicon substrate when charge is applied to, or removed from, a transistor's gate. This flow of electrons consumes power and produces electromagnetic radiation.

The power consumption of an integrated circuit or a larger device reflects the aggregate activity of its individual elements, as well as the capacitance and other electrical properties of the system. For example, a microprocessor may use a different circuit to dispatch an addition operation than a register load, causing these operations to consume different amounts of power. Net power consumption depends also on which transistors are switching within the active circuits. Some transistors' activity depends on the data the circuit is processing. For example, more transistors may switch when adding the hexadecimal bytes A7 to B9 than when adding 01 to 00.

Because the amount of power used by a device is influenced by the data being processed, power consumption measurements contain information about a circuit's calculations. Even the effects of a single transistor, while not directly observable in power measurements from a large devices, do appear as weak correlations. When a device is processing cryptographic secrets, its data-dependent power usage can expose these secrets to attack.

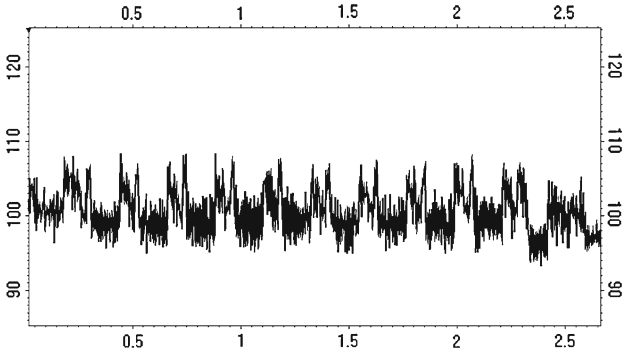

Fig. 1 Power trace from a smart card performing an AES-128 encryption, with the ten rounds clearly visible

2.1 Traces and frequency distributions

The first step in the power analysis process is to collect one or more *traces* from the target device. A trace is a sequence of measurements taken across a cryptographic operation or sequence of operations.

Figure 1 shows approximately 3 ms of a power trace collected from a smart card performing an AES-128 encryption operation. The power consumption was sampled at 100 MHz, and each point in the trace is the average of multiple samples. The trace data were captured by placing a resistor in series with the device's ground line, then using an oscilloscope to measure the voltage at the ground input. Accordingly, a larger measurement (higher values on Fig. 1) represent higher power consumption.

Figure 2 shows a 5 microsecond segment of a power trace recorded from an FPGA encrypting 1 MB of data using AES-128 in CBC mode. Four individual AES-128 encryptions are visible in the figure. The full trace recorded nearly 40 million measurements at 500 MHz, spanning the 2^{16} AES operations in the complete CBC encryption. This trace was captured by placing a resistor in series with the power (Vcc) input and measuring the voltage at the device, so higher power consumption appears as a *lower* value in the figure.

Although the AES rounds are clearly identifiable in both Figs. 1 and 2, clean measurements such as these are *not* required for DPA attacks. Further details of the analysis setup and data collection will be discussed in Sect. 2.4.

The following experiment, conducted using a set of traces collected from the smart card, illustrates how power consumption can be dependent on sensitive data. Figure 3 shows

[1] We have chosen to introduce DPA before simple power analysis (SPA) because DPA is the more important subject. SPA also lacks the noise-filtering properties of DPA, so readers who are first exposed to SPA may find DPA to be counterintuitive.

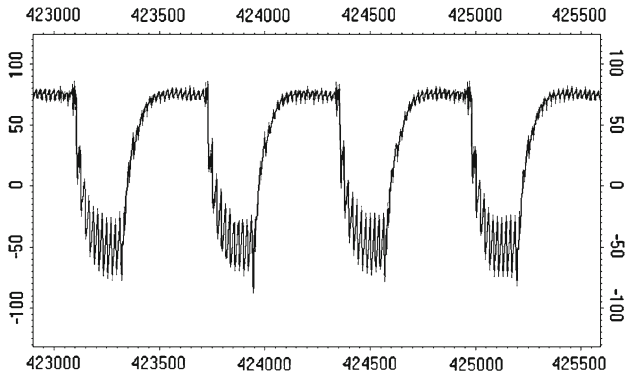

Fig. 2 Part of a power trace from an FPGA performing AES-128 CBC mode encryptions. In power side measurement, down corresponds to more power

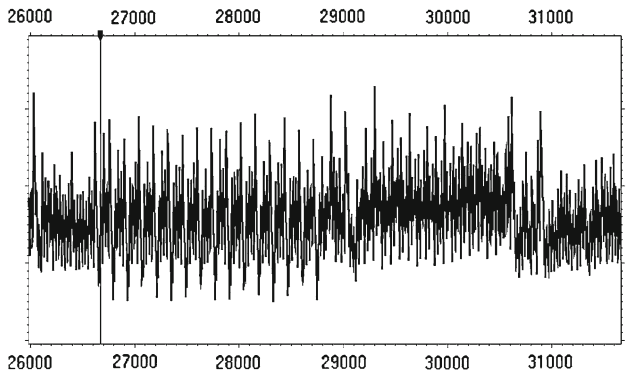

Fig. 3 Power trace segment showing the first round of AES-128 encryption on a smart card. A vertical line marks the location of first S-box lookup

the region of the power trace from Fig. 1 during the first round of the AES-128 operation. (Figure 3 shows more detail than Fig. 1 because fewer points are averaged together to produce each point in the plot.) The moment in time when the card computes the output of the first S-box is marked by a vertical line.

A set of 4,000 traces were captured from the same smart card while performing AES-128 encryption operations. Each trace observes the encryption of a different, randomly chosen, plaintext. The same random known key was used each time. Figure 4 shows the distribution of power measurements, at the time marked in Fig. 3, among the 4, 000 traces. An 8-bit A/D converter was used, so the possible range of points is from 0 to 255. The observed distribution is close to a Gaussian distribution, with mean of 120 units and standard deviation of 10.66.[2] Figure 4 shows that there is significant variation in the power consumption measurements among the traces at

[2] We chose a Gaussian approximation to simplify the exposition. In general, Gaussian mixtures are a better fit for the power consumption distributions observed in practice. For more details see Chapter 4 of [10].

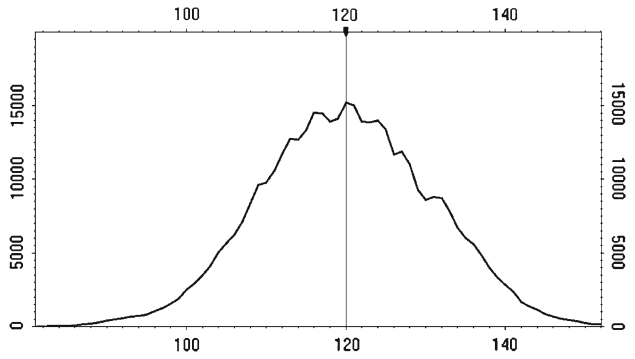

Fig. 4 Distribution of power consumption at first S-box output computation

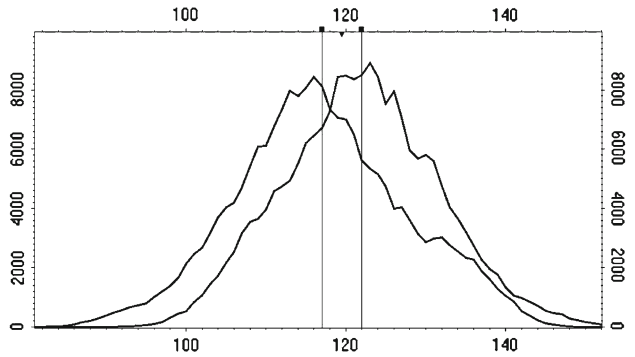

Fig. 5 Distributions of power consumption measurements for traces with the LSB of the output of the first S-box being 1 (left) and 0 (right)

this point during the computation. Such variations combine a range of effects, including data-dependent variations in the cryptographic processing, other activity in a device, measurement inaccuracies, interference, environmental factors, and so forth.

Figure 5 confirms that data-dependent power consumption contributes to the variation observed in the power traces. The figure shows two distributions of power measurements. The distribution toward the left was produced using only the traces where the least significant bit (LSB) of the output of the first S-box is 1. The distribution toward the right was produced using only the traces where the LSB is 0. In this case, the key and the plaintext were known, so the S-box output could be computed from the plaintext.

For the traces where the LSB was 1, the power consumption was approximately Gaussian with mean 116.9 and standard deviation 10.7. When the LSB was 0, the distribution was approximately Gaussian with mean 121.9 and standard deviation of 9.7. The placement of inverters by logic synthesis tools and other design details make it possible for either value to consume more power. What matters is that these two distributions are significantly different, demonstrating that the power consumption is statistically correlated to the LSB of the S-box output. The distributions in Fig. 5 overlap

significantly, so a single measurement will not be sufficient to determine the value of the S-box output bit, but these distributions can be reliably distinguished given a sufficiently large set of measurements.

The subsets shown in Fig. 5 are not the only way to divide the data in Fig. 4. For example, if the data are divided into two subsets based on the value of any other output bit from the first S-box, the resulting distributions are also clearly distinguishable. This reaffirms that the power consumption at the selected point in time is dependent on all bits of the output of the first S box.

2.2 Differential power analysis

Differential Power Analysis DPA is a statistical method for analyzing sets of measurements to identify data-dependent correlations. The basic method involves partitioning a set of traces into subsets, then computing the difference of the averages of these subsets. If the choice of which trace is assigned to each subset is uncorrelated to the measurements contained in the traces, the difference in the subsets' averages will approach zero as the number of traces increases. Otherwise, if the partitioning into subsets is correlated to the trace measurements, the averages will approach a nonzero value. Given enough traces, extremely tiny correlations can be isolated—no matter how much noise is present in the measurements.

Recall that Fig. 4 showed the probability distribution of measurements at a particular offset in a set of traces. Figure 5 showed the two component distributions for the cases where the cryptographic calculation happened to produce a 0 or a 1 in the least-significant bit (LSB) of the first S-box lookup. In other words, when the data points in Fig. 4 are correctly divided into subsets according to the value of the LSB of the output of the first S-box, the difference of the subsets' averages will converge to the difference of the means of the distributions in Fig. 5. Even if the difference of the means is very small (such as the effect of a single transistor within one chip in a complex device), the difference will eventually become statistically significant given a sufficient number of traces. On the other hand, if the partitioning into subsets is uncorrelated to the data being processed, then each subset is essentially a random sampling of measurements from the full distribution shown in Fig. 4 and, except for sampling errors, the subsets' distributions will be the same as the full distribution. As the sample size increases, the sampling errors will diminish, so the subsets' averages will converge to the mean of the full distribution and the difference of the subsets' averages will converge to 0.

Figure 5 only reflects measurements at one carefully chosen point in time during the operations. In practice, the location of greatest leakage may not be known prior to the analysis. To avoid the need for information about the target

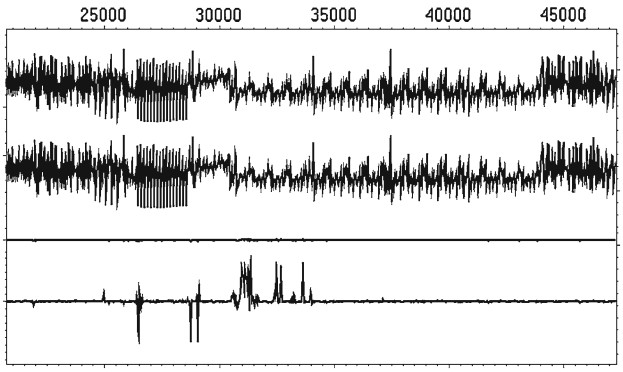

Fig. 6 Typical DPA result showing (from top to bottom) the average of the traces where the LSB of the output of first S-box in round 1 is 1, the average of traces where the LSB is 0, the difference between the top two traces, and the difference with the Y axis magnified by a factor of 15

device, the averaging process for DPA is typically performed at a range of offsets within the traces. The basic DPA process examines the difference of these averages at each point in the set of traces. The DPA result can be graphed where the X axis is the trace offset (time) and the Y axis shows the difference in the averages of the two distributions at that point. At offsets where the power consumption is correlated to the selection function output (e.g., because the device is manipulating this value internally), the distributions at that offset will differ, resulting in a nonzero value (e.g., a spike) in the graph. In regions of the graph where the power consumption is unrelated to the selection function output, the distributions will not have statistically significant differences. The points in the graph will converge to 0 as the number of traces increases, making these regions appear flat.

Figure 6 shows the components of a typical successful DPA result. In this case, the target device is the same smart card performing AES, and the subsets are based on the actual values of the LSB of the first S-box output in the first round. Four traces are shown in the figure. The uppermost trace is the average of the traces for which the LSB was 1 over the time interval covering the first two rounds of the AES encryption. The second trace is the average of the traces for which the LSB was 0. The top two traces appear to be the same, since the difference between the averages is much smaller than overall power consumption variations. The third trace shows the difference of the top two traces, and appears mostly flat—again, because the differences are small. The fourth (lowest) trace shows the difference of the averages with the Y axis scaling increased by a factor of 15, and the DPA results are clearly visible.

Areas of leakage are visible as spikes in the lowest trace. The first spike occurs when the S-box output bit is first computed by the target device. Further spikes appear when this bit is further processed in the rest of the first round. By the

end of the first round, the AES intermediates are no longer correlated to the LSB of the S-box output due to mixing with other bits and the cipher's avalanche. As a result, no spikes are seen in the difference trace in the second round or beyond. Figure 6 shows relatively little noise between the spikes, since the number of traces used (4,000) is high relative to the noise within individual traces. While more traces yield cleaner results, there is normally no reason to use more data than is necessary to distinguish the desired signals.

The information revealed by a DPA test is determined by the choice of *selection function*. A selection function is used to assign traces to subsets and is typically based on an educated guess as to a possible value for one or more intermediates within a cryptographic calculation. If the final DPA trace shows significant spikes, the cryptanalyst knows the selection function output is correlated to (or equals) a value actually computed by the target device. If no correlation is observed, then selection function output was not correlated (or the correlation was too small to observe). Selection functions may be the predicted value of a single bit, such as an output bit from an S-box or multiplier. More complex functions, such as the predicted difference between the value of a bit in a register and the value of a bit that overwrites it, may also be used. Selection functions can also be functions of multiple bits. For example, a selection function might output 1 if a multi-bit intermediate is predicted to equal a constant (or a different intermediate) and otherwise output 0. As we will describe later in Sect. 4.2, the choice of selection function is a key part of the DPA process as it encapsulates the engineering intuition about leakages that may be present in a device. Selection functions used in DPA are typically binary valued functions.

2.3 Using DPA to attack AES: an example

To illustrate the DPA process, we now present typical a DPA attack on AES-128 encryption using the AES-128 smart card traces.[3] As previously shown in Fig. 5, computational intermediates, such as the outputs of the AES S-box, have a small statistical influence on power consumption measurements. We now describe how these small correlations can be used to reveal the secret key.

The first round of AES-128 encryption consists of the following steps:

1. *Initialization* The initial 16-byte state of the cipher, organized as a 4×4 byte matrix, is initialized to the 16 bytes of the plaintext.

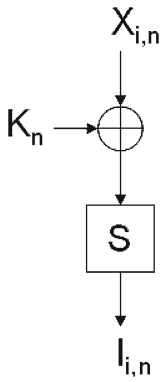

Fig. 7 AES S-box lookup during first round

2. *AddRoundKey* The 16-byte secret key is exclusive-ORed with the 16 bytes of the plaintext state.
3. *SubBytes* Each byte of the state is replaced by another using the S-box, which is an invertible lookup table.
4. *ShiftRows* Bytes in each row of the state are shuffled.
5. *MixColumns* Each column of bytes of the state is mixed using a linear operation.

The DPA attack will target the output of AddRoundKey and SubBytes in AES. These operations are shown in Fig. 7.

For each trace i, Let I_i denote the 16-byte intermediate state of the cipher just after the SubBytes step in round 1. Let the nth byte of this state (where $n \in \{0, \ldots, 15\}$) be denoted by $I_{i,n}$. Let the first round key be denoted by K and its nth byte be denoted by K_n. Similarly let $X_{i,n}$ denote the nth byte of plaintext X_i used for the ith trace. As shown in Fig. 7, $I_{i,n}$ only depends on one byte $X_{i,n}$ of the input and one byte K_n of the key, i.e.,

$$I_{i,n} = S[X_{i,n} \oplus K_n] \tag{1}$$

In this equation, $X_{i,n}$ is a **known variable**: one byte of plaintext. K_n is a **secret constant**. S is the AES substitution table, which is defined in the AES standard, $I_{i,n}$ is therefore an **unknown variable** which depends on a 1-byte secret constant and other known quantities.

AES can be broken easily if there is an efficient test that reveals whether a given candidate for K_n is correct. In particular, K_n is an 8-bit value, so at most 256 queries of this test would be required to confirm the correct K_n. The 16 K_n bytes that make up the entire AES-128 key could be found by simply solving for each byte separately.

DPA provides a practical way to test if a candidate value of K_n is correct. The candidate K_n is used with equation (1) to derive the value of $I_{i,n}$ for each trace's $X_{i,n}$. A selection function can be developed based on the calculated $I_{i,n}$. In this example, bit 0 (the LSB) of $I_{i,n}$ was used as the selection function output.[4] Each trace is assigned to one of two subsets,

[3] The initial discussion is using the same device for consistency, not because there is anything about the attack that is specific to smart cards or this particular device. We will discuss attacks against other devices later, e.g., see Fig. 9.

[4] For many devices, such as the one in Fig. 6, all bits will work well. For some devices, however, different selection function choices may yield stronger correlations.

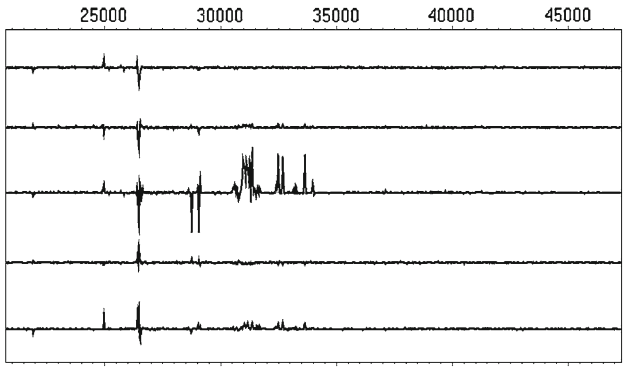

Fig. 8 Five differential traces for the DPA test predicting the LSB of $I_{i,0}$ for guesses $K_0 = 101, \ldots, 105$ from top to bottom, with the correct key $K_0 = 103$, corresponding to the third trace

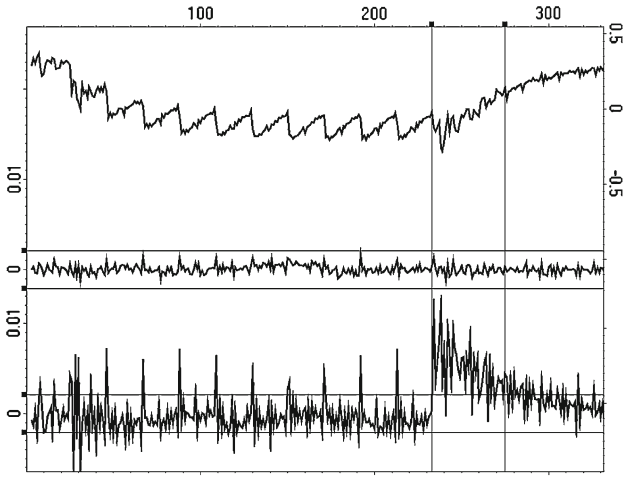

Fig. 9 DPA results showing the average trace for an AES-128 operation running on an FPGA (top), the differential trace for an incorrect guess of a byte of the last round key (middle) and the differential trace for the correct key byte (bottom)

depending on whether the selection function result is 0 or 1 for the candidate K_n and the plaintext being encrypted when the trace was captured.

The difference of the subsets' averages is then examined. If the value of the S-box output bit predicted by the selection function has even a tiny correlation to the power traces, the DPA test will show spikes indicating that the candidate K_n is correct. For each wrong K_n, the predicted values of $I_{i,n}$ will be (largely) unrelated to any data being processed by the target device, and the DPA test will not be (or will be much less) statistically significant.

Difference traces were prepared for all 256 possible values for K_0 (i.e., $K_0 = 0, \ldots, 255$). Figure 8 shows, from top to bottom, five traces for $K_0 = 101, \ldots, 105$. The correct value for K_0 is 103, as is obvious from the presence of large spikes in the $K_0 = 103$ trace (which matches Fig. 6). Traces for incorrect K_0 values have much smaller spikes [5] or are relatively flat.

The same analysis can be repeated for all the 16 bytes of the state ($n = 0,\ldots,15$) to recover the entire 128-bit AES secret key from the device. The same traces can be reused in finding each key byte; it is not necessary to collect separate data, since each test is checking for different correlations in the data set.

A DPA test can be summarized as follows: Let T denote the set of traces that are collected and let T_i denote the ith trace. Let $T_i[j]$ denote power measurement or sample at the jth time offset within the trace T_i. Let C denote the set of known inputs or outputs for the traces with C_i corresponding to the ith trace. Let $D(C_i, K_n)$ denote a binary valued selection function with input C_i and the guess K_n of a part of a key. Each point j in the differential trace Δ_D for the guess K_n is computed as follows:

$$\Delta_D[j] = \frac{\sum_{i=1}^{m} D(C_i, K_n) T_i[j]}{\sum_{i=1}^{m} D(C_i, K_n)} - \frac{\sum_{i=1}^{m} (1 - D(C_i, K_n))\, T_i[j]}{\sum_{i=1}^{m} (1 - D(C_i, K_n))}$$

For a typical DPA analysis, the guess for K_n that produces the largest spikes in the differential trace Δ_D is considered to be the most likely candidate for the correct value.

The attack can be adjusted easily for other cipher modes and target devices. For example, Fig. 9 shows a DPA result from the FPGA implementation of AES-CBC shown in Fig. 2. For convenience, a single oscilloscope capture was used to capture all AES operations needed for the attack, then the capture file was divided into 65,536 separate AES operations for analysis. Also, because the ciphertext (instead of plaintext) was available, the DPA process was used to find bytes of the last round key.[6] The top trace in Fig. 9 is the average power trace for an AES operation. The middle trace is a differential trace for a DPA test carried out with an incorrect guess for the first byte of the last round key and the bottom trace shows the corresponding differential trace for the correct key byte guess.[7]

[5] The presence of smaller spikes for incorrect hypothesis is due to harmonics, which are discussed in Sect. 4.4.

[6] This analysis is a ciphertext-only attack; knowledge of the plaintext is not required.

[7] Although the FPGA yields less information *per AES operation* than the smart card in Fig. 1, the FPGA leaks its key out more quickly because it performs many more AES operations per second. The FPGA analysis was automated using a simple automated tool to identify and synchronize the individual AES block operations. The entire process (including the time for the FPGA to perform the AES operations, the capture and transfer of the trace data to a PC, and all necessary processing and averaging steps to solve for the complete key) took 125 s from start to finish.

2.4 Stages of a black-box DPA attack

A typical DPA attack involves the following stages:

- *Device instrumentation* This preliminary stage involves developing the means to communicate with the device to invoke cryptographic operations and to record its responses. The measurement apparatus, such as digital oscilloscope driven by a PC, is also connected to the target device. Depending on the device and the access available, a resistor or a current probe in series with the device's power or ground lines can be used. Measurements taken closer to the cryptographic component will generally be of better quality, although a larger number of lower-quality traces can also be used. If a resistor cannot be inserted (e.g., if a device uses an internal battery), the device's internal resistance is often sufficient. E-field and M-field probes can also be used to conduct EM attacks on a device, using the same methodology as DPA. For triggering, the measurement system is typically connected to the device's I/O lines.
- *Measurement* This is the data collection stage. Power traces are recorded while the target device performs cryptographic operations. Each captured trace is stored on a PC with the associated cryptographic data (e.g., the plaintext or ciphertext). As needed, trace quality and capture efficiency may be improved by adding analog filters, adjusting bandwidth or sampling rates, and by exploring SPA signal characteristics (see Sect. 4.1) to remove irrelevant regions.
- *Signal processing* This optional stage involves processing traces in software to remove alignment errors, isolate features of interest, highlight signals, and reduce noise. In many cases, only simple temporal alignment will be necessary, or this step can be omitted entirely.
- *Prediction and selection function generation* In this stage, different hypotheses about a portion of the key formed are used to define selection functions for analysis. Each selection function is then applied to the cryptographic data associated with each trace, deriving a prediction about an intermediate state for the next stages to test.
- *Averaging* The averaging stage computes, for each selection function, the averages of the input trace subsets defined by the selection function outputs. This step is normally the most computationally intensive stage.
- *Evaluation* The DPA test results are analyzed to determine the most likely candidate key guesses. This process can be done visually or using automated tools.

The final three steps (prediction, averaging, and evaluation) are often iterated. For example, with AES-256, the first round key is typically found before the attack can begin on the second round key. In other cases, additional steps may also be repeated, e.g., if adaptively chosen input messages are being used.

3 Simple power analysis

This section introduces SPA. In [11], SPA was described as "a technique that involves directly interpreting power consumption measurements collected during cryptographic operations. SPA can yield information about a device's operation as well as key material." SPA exploits major variations in a power consumption. As a result, unlike DPA, the method is generally unable to extract keys from noisy measurements.[8] For many devices, however, SPA provides a very effective and efficient way to obtain the information necessary to solve for the secret keys.

3.1 SPA methods

Simple power analysis is a collection of methods for inspecting power traces to gain insight into a device's operation, including identifying data-dependent power variations. SPA focuses on examining features that are directly visible in a single power trace or evident by comparing pairs of power traces.

This section introduces the common SPA analysis methods, including single trace analysis and trace pair analysis.

3.1.1 Single trace analysis

Within a single power trace, features which correspond to timing, device attributes, algorithm structure, or other properties of the computation are often visible. The first stage of simple power analysis involves looking at a power trace and drawing inferences about the operation.

For example, Fig. 10 was captured from a smart card running a pseudorandom number generation operation using 3DES and an EEPROM. Variations in power consumption convey information about the device's operations. From left to right, the trace shows the arrival of input data to the device, the output of a single byte from the device, a triple-DES operation, and a series of EEPROM writes. Some of these suboperations, such as the I/O and EEPROM writes, can be recognized from their timing and power consumption profiles, or using background knowledge about the device and the protocols it implements. In other cases, information

[8] Some methods fall in a gray area between SPA and DPA or combine elements of both. For example, if individual traces are of low quality many oscilloscopes can internally average power traces captured from repeating the same operation several times, providing a higher-quality trace for analysis using SPA techniques.

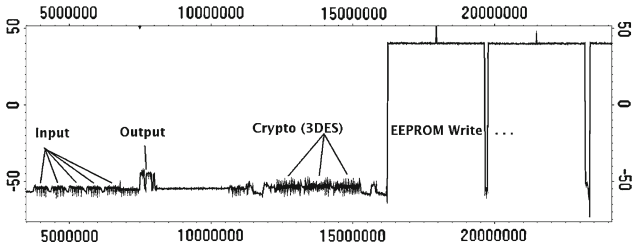

Fig. 10 Power trace from a smart card that is performing a 3DES-based PRNG operation

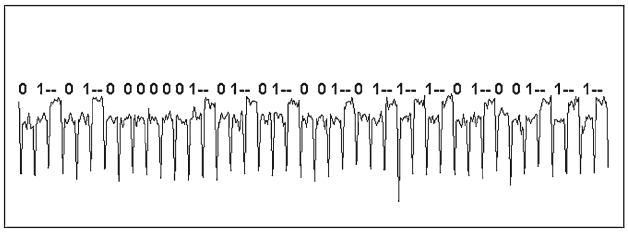

Fig. 11 SPA leaks from an RSA implementation

about what the device is doing can be inferred by recognizing repeated patterns and counting iterations in loops.

Different information is available at different levels. The view in Fig. 10 is zoomed out show each of the major operations involved in the transaction. Three bursts of activity are noted in the portion labeled "Crypto (3DES)". Zooming in on the first of these bumps reveals a pattern repeated 16 times—which is the DES round function. If the analyst did not know that the protocol was using 3DES, structural clues such as these can be helpful. Zooming in to the clock cycle level can reveal more detail, such as low-level implementation choices.[9]

While two segments may look similar to the eye, a more reliable method of identifying subtle differences is by computing the difference between two segments of a trace. The approach usually used here is to make a copy of the trace, shift it by some time interval and then compute and display the point by point difference. When two segments of a trace are truly similar, the difference between them should be relatively flat over the range where they agree.

SPA leaks that are evident in a single trace can also reveal cryptographic secrets. Figure 11 shows a segment of the power trace of a modular exponentiation loop in which direct interpretation of the SPA features reveals an RSA decryption key.

[9] For example, in this implementation, the C and D rotations in the key schedule use a software routine that rotates 26-bit quantities by one bit. The routine is called once in rounds 1, 2, 9, and 16 and twice in the other rounds of the DES encryption. The timing variation that this creates—although it does not leak any information about the secret key—is a distinctive SPA signature.

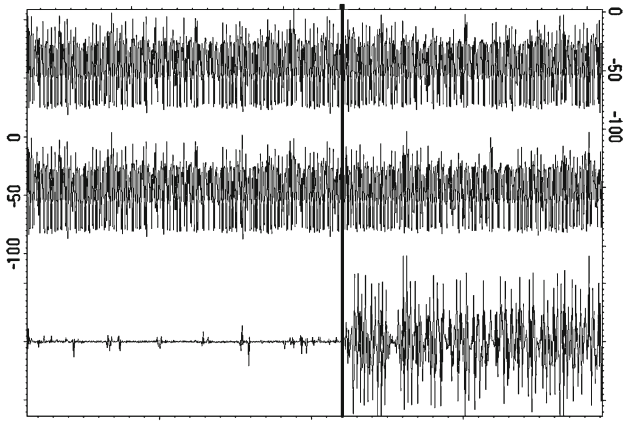

Fig. 12 Two traces and their difference, with point of divergence indicated

This trace shows a sequence of squares and multiplications as the device performs modular exponentiation using the binary left-to-right algorithm. Multiplications consume more power than squares in this trace, and appear as by higher peaks. In the binary left-to-right algorithm, one square is performed in every iteration of the exponentiation loop, while multiplications are only performed when a bit of the exponent is 1. This fact allows the pattern of operations in Fig. 11 to be interpreted. Each 1 bit in the secret exponent appears as a shorter bump followed by a taller one, while a 0 bit appears as a shorter bump without a subsequent taller one. The bits of the exponent can thus be recovered as shown.

3.1.2 Trace pair analysis

Trace pair analysis involves comparing traces to identify similar regions and differences.

Figure 12 shows two power traces recorded from a device performing the GSM Authenticate Subscriber command using the A3 algorithm. The device used the same key each time, but with different initial data values, yielding the upper two traces. The lower trace shows the difference between the top traces.

The traces are aligned at the left edge of Fig. 12 and stay synchronized across the first half of the figure—and the difference trace is relatively flat. At the point indicated by the dark vertical line, however, the traces diverge and large differences appear, indicating that the computations have taken separate paths.

Figure 13 shows the same two traces, but the second trace has been shifted 21 clock cycles (4.2 μS) to the left to bring them back into synchronization at the location of the rightmost vertical black line. The black line from Fig. 12 has been broken, and still marks the points that were originally aligned.

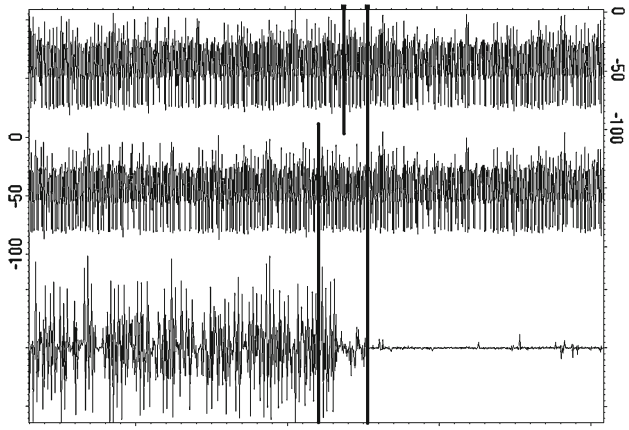

Fig. 13 he traces of Figure 12, aligned to showing point of rightward synchronization, and the corresponding difference trace. The original point of divergence is marked by the first, broken line

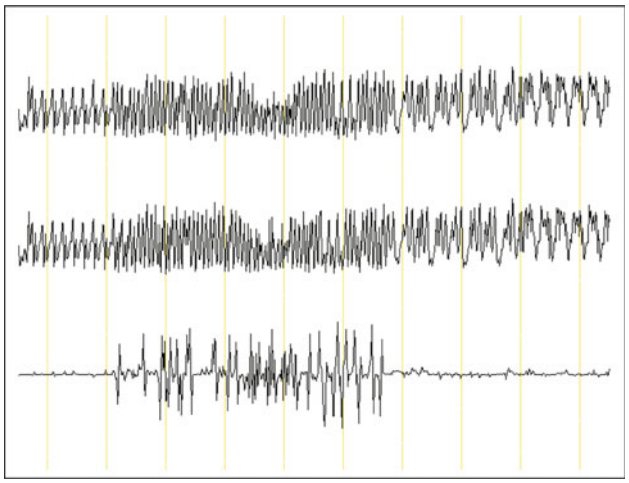

Fig. 14 Trace pair comparison of a permutation with different inputs

Figures 12 and 13 together indicate that the two calculations briefly took different paths, with the divergent region taking 20 clock cycles in the upper trace and 41 clock cycles in the second trace. This characteristic was caused by a conditional branch that took different execution paths in the two traces.

SPA leaks that result in timing differences can often be attacked using timing attacks [5], but SPA generally provides more information than overall timing. Figure 14 shows traces from a permutation function being run with two different input messages. In this case, even though the total operation time is the same, high-amplitude leaks are evident. Once the cause of the variations is well understood, leaks such as these may provide information that can be used to recover keys.

In analyzing pairs of traces, macroscopic SPA features in the two traces allow similar features to be aligned and compared. The simple method of subtracting one trace from the other highlights amplitude and timing differences between them.

3.2 SPA Leaks

Data-dependent conditional branches are one source of SPA leak. Another common source of SPA leaks is CPU instructions with variable timing, such as multiplications on the ARM7 and Intel 80486. Even in branchless code with constant timing, instructions that have microcode variations can have visible data-dependent variations in power consumption. Arithmetic and multi-precision integer operations can have major variations in computation complexity that lead to leaks having high amplitude variations. An SPA leak may be triggered by a low-probability event. For example, a multi-precision integer multiplication may run faster for each 16-bit word of its input that is zero. SPA variations unrelated to cryptographic processing, such as timer interrupts and context switches on multithreaded CPUs, are also common but these tend to be less useful and can often be recognized because they are not consistent for a given key and data input.

Specially chosen inputs can be used to search for SPA leaks corresponding to unusual intermediate states. Additional trace pairs can be examined to determine whether characteristics are consistent when the key and data are each modified or held constant. As will be discussed later, SPA leaks depending on the key or on computation intermediates tend to be particularly useful for cryptanalysis.

3.3 SPA attacks

Once an SPA leak is identified, the next step is to use it to recover the key. In some cases, such as the device in Fig. 11, the process is simple. In other cases, more analysis may be required.

The traces in Fig. 15 show an SPA leak that appears in an RSA decryption implementation that uses the Chinese Remainder Theorem (CRT). The figure shows the final CRT processing steps from decrypting the two messages. The ciphertexts were chosen to have about half of the most-significant bits of the plaintext set to 0.[10]

The bottom trace shows a bump that appears for messages larger than a threshold near 2^{512}. The top trace shows the same region when M is smaller than the threshold. Further testing showed that this SPA leak reveals whether $M \mod p > M \mod q$. Using an adaptive chosen ciphertext attack, the threshold $M = q$ can be located by binary search, revealing the RSA private key. A similar attack was described in [12].

[10] The plaintext values were selected then encrypted using the public key to form the ciphertexts sent to the device for decryption. Although the padding is invalid, the private key operation is completed before the padding can be checked.

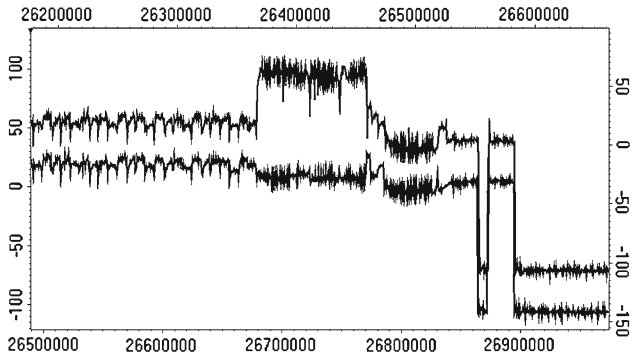

Fig. 15 A trace from a device performing RSA using CRT for which the decryption result is less than q, and a trace with decryption result greater than q

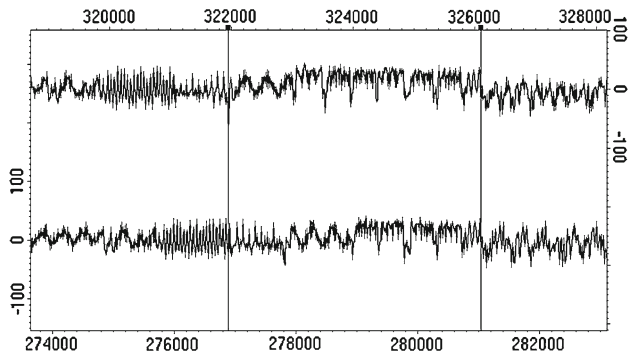

Fig. 16 Trace pair showing branch difference that reveals one bit of the nonce during an EC-NR signature

In black box evaluation, it may be harder to directly interpret the meaning of an observed leak. **Collision attacks** can be a relatively simple way to exploit SPA leaks [13,14]. In particular, SPA can be used to infer when pairs of inputs lead a device into colliding (or similar) states—without requiring a precise understanding of the nature of the device's leakage. In some cases, the ability to identify these inputs enables recovery of the key.

Algebraic attacks and specifically **lattice-based methods** are another avenue for exploiting SPA leaks that give just a few bits of information per trace.

For example, Fig. 16 shows two power traces from a device performing an ECC curve multiplication using a nonce. For this device, 20 bits of the nonce could be easily inferred from an SPA leak per multiplication on a 168-bit elliptic curve. The top trace shows a nonce bit equal to 0, and the bottom trace shows an operation with the nonce equal to 1. The value of the nonce bit has a clearly visible effect in the region of the traces between the vertical lines. Using a lattice attack based on work by Bleichenbacher [15] and others (see [16–18]), the signing key can be recovered easily. Algebraic attacks have also been shown to be effective against implementations of symmetric cryptographic algorithms [19,20].

As illustrated by these examples, recovery of the secret key is often straightforward once an SPA vulnerability is identified and characterized. This is not surprising, since the adversary has useful information about computational intermediates that the cipher designer assumed would be hidden.

SPA is only practical when significant, data-dependant features in the power traces are apparent. In practice, data-dependant power variations during cryptographic computations may be hidden in noise. In other situations, SPA leaks may be visible, but their interpretation is so tedious that an automated attack is more attractive. For these situations, the statistical power and structural simplicity of a DPA attack are advantageous.

4 Implementing DPA

Attackers and product evaluators are motivated to obtain keys as quickly and easily as possible. If basic DPA is not immediately successful, there are many ways to adapt the attack to compensate for countermeasures or to reduce data collection or processing time. In many cases, these adjustments are not essential, but are helpful for the impatient adversary. This section elaborates on the DPA testing process and techniques described in Sects. 2.2 and 2.4. These adjustments are based on our experience in analyzing a broad range of algorithms and devices.

4.1 DPA: Data collection and preparation

Leaks exploited by DPA can be much smaller than the level of noise in a set of traces, but better signal-to-noise ratios require fewer traces. Time is often of the essence in an evaluation lab or adversarial setting, and improvements in the initial data collection can reduce the time to recover keys by improving signal quality.

4.1.1 Device instrumentation

Many factors influence the signal quality of power measurements. For example, taking measurements closer to the crypto IC typically improves the traces. On large ASICs/SoCs with multiple power and ground connections, inputs that power the circuitry that performs the cryptographic computations are likely to provide better data. Similarly, at the board level, removal of decoupling capacitors or use of an external bench-grade power supply can reduce noise.

While a resistor in series with a power or ground line is the simplest way to obtain power traces, we have also had success exploiting the internal resistance of batteries and internal power supplies. If direct power traces are unavailable or of poor quality, other sensors can be used instead. For example, magnetic field pickups can be effective for larger ICs. While thermal imaging and acoustic effects have been suggested as possible side channels [21,22], the quality of the resulting

measurements is likely to be low. If the integrated circuit has been decapped, photon emissions measurements [23,24] and other side channels may be options.

In some cases, signal quality is affected by a device's operating parameters, such as voltage, temperature, and clock rate/waveform. Stressing a device by running it near the edge of its operational envelope may enhance the leak being targeted or reduce the effectiveness of certain countermeasures. For example, lowering the input voltage may stress voltage regulators and increase leakage. A bench-grade clock can reduce timing jitter, especially when synchronized to the sampling clock of the measurement apparatus. Using a sine-wave clock may reduce high-frequency noise in measurements.

For some devices, it is possible to control the number of cryptographic operations performed in a given command, e.g., by selecting the input message length. Commands that perform more cryptographic operations are often preferred, since increasing the number of cryptographic operations per trace usually speeds up the data collection stage. For example, the single CBC-mode encryption trace from Fig. 2 which contained 2^{16} AES encryptions was collected in a few seconds, whereas collecting 2^{16} power traces of single AES encryptions on the same device takes over an hour due to the communication and data transfer overheads.

While many DPA attacks work with random or arbitrary input messages (such as typical ciphertext), chosen input messages can sometimes reveal additional leakage or simplify the analysis. For example, holding part of the input constant can enable a practical attack [25,26] or decrease the complexity of subsequent steps on the analysis [27,28]. Holding part of the input constant can also amplify leakage by increasing the number of intermediate bits that are correlated to the value being predicted. Depending on the algorithm being analyzed and the attack strategy, it may be most efficient to use a sequence of adaptively-chosen input sets, where each set of inputs depend on the results of a DPA analysis done on the prior set. In other cases, such as with the "doubling attack" [29], a chosen message strategy can help circumvent poorly-designed DPA countermeasures. In some instances, when inputs are not fully controllable by an attacker, fault or glitching attacks may be used to influence the cryptographic operations being performed. Adaptively chosen inputs have also been used in the context and timing analysis [30], and the efficiency of such side attacks has been formally modeled and analyzed in [31,32].

4.1.2 Measurement

Data collection is performed with a high-speed analog-to-digital conversion system. Digital storage oscilloscopes are well-suited to this task. In selecting a scope, reasonably deep memory (for capturing longer traces) and trigger flexibility (to help start trace capturing at the appropriate time) are helpful. In addition, rapid trigger re-arming time and fast transfer rates can help speed up the data collection phase. (Data collection is often the most time-consuming step of DPA.)

In our experience, signal fidelity and calibration are less important, since the types of distortions introduced by cheaper lab equipment generally do not interfere with the leakage signals exploited by DPA. While it might appear that sampling resolution would matter significantly when dealing with very small correlations, the primary concern for DPA is the signal-to-noise ratio, and sampling errors are usually significantly smaller than other noise sources.[11]

Analog pre-processing of the raw signal prior to A/D conversion is helpful in some situations. For example, in the case of field-powered devices and some types of EM analysis, an AM or angle demodulation step helps isolate the signal of interest. Similarly, simple bandwidth limiting (a feature in many oscilloscopes) can help remove unwanted high-frequency artifacts.

To minimize the amount of extraneous data collected, it may also be helpful to examine any SPA signals to help narrow down when the cryptographic process occurs (see Sect. 3).

In other cases, a preliminary DPA test using the known input and output bits as selection functions can highlight the precise location where the cryptographic operation is performed. A more general technique for characterizing the leakage from a device is to perform DPA tests using selection functions based on the expected intermediate values within the cryptographic operation. This "known-key" analysis is only possible if the attacker can obtain a device with a known key. Tests using these intermediates help identify what information is leaking from the implementation. For example, Fig. 6 used known key analysis to illustrate how the least-significant bit of the first S-box lookup leaked from an AES implementation.

4.1.3 Signal processing

After traces have been collected in digital form, additional digital signal processing can significantly improve the efficiency of the rest of the DPA process and improve its outcome.

Trace alignment (identifying a reference time location in each trace) is typically performed and is the simplest signal processing technique. To align a trace T_i against a reference

[11] A biased coin provides a more familiar example of how a signal can be identified with precision exceeding the measurement system. Given enough measurements, the coin's bias can be determined with arbitrarily fine accuracy—even though the individual measurements each have only one bit of resolution (heads or tails).

trace T_0, a simple correlation test is employed to find the time shift d that minimizes the differences (or the square of the differences) between $T_0[j]$ and $T_i[j+d]$. Occasionally, more complex alignment methods are needed. For example, clock drifts across traces can be a source of misalignment, and several countermeasures have been designed to create misalignments. (These include shuffling the order of operations, insertion of random no-ops or clock skips, and use of desynchronized clocks.) Often, careful alignment or signal resynchronization can reduce or eliminate these effects (see [33] and Sect. 6). Although correct alignment is *not* necessary for DPA to succeed, good alignment does reduce the number of traces required to extract a key.

An alternative to performing alignment is to perform DPA in the frequency domain. To do this, the signal processing phase involves performing Fourier analysis on the relevant regions of each trace.

Other digital filtering of signals at this stage can also help reduce noise and to focus on the parts of the spectrum where the leakage signal is present. For example, trace "compression" can be performed by adding together successive measurements, and can help reduce high-frequency noise and amplify signal resolution while reducing the amount of data that requires processing in subsequent steps. If unwanted repetitive effects are present, these can be detected and subtracted from the traces. Another simple strategy to reduce extraneous and measurement noise from traces is to collect multiple traces while an identical operation is repeated, then average these together.

DPA testing can be a highly data intensive task, especially when performed with a very large number of traces each containing a large number of measurement points. Much of the information present within the traces is not useful in the DPA test; only a few trace offsets typically show DPA signals. Once a particular device has been characterized, traces can be greatly compressed by discarding all points except the few that matter.

At this stage, traces may also be prepared for analysis by variants of DPA, such as higher-order DPA (see Sect. 5). In the simplest case, certain traces may be discarded (filtered out) based on the value of some portion of the signal. For example, if all traces with below-average measurements at a first location are discarded, high-order effects involving this location and any other location will appear. More generally, a function can be applied to transform n-tuples of measurement points as part of an nth-order HODPA attack. A distortion function may be applied to traces to enhance a ZO-2DPA attack [34].

4.2 DPA: Hypothesis Generation Using Selection Functions

DPA attacks exploit leaked information by leveraging a prediction about some aspect of the computation that varies in a key-dependant manner. Section 2.2 gave an example of an attack targeting an intermediate bit in the first round of an AES encryption. Many aspects of the device state may leak, for example, "Hamming distance" leaks correlate to the number of bits that change when the value of the word on a bus or in a register changes. Real leaks can be complicated. In some devices, transitions from 0 to 1 leak differently than transitions from 1 to 0. Hamming weight and Hamming distance models treat bits independently, but in real devices, the amplitude of leaks varies significantly for different bits, and may be further modulated by multi-bit effects. Some devices have word-oriented leaks which may be correlated to flag bits such as sign, zero, overflow, or carry.

On the SASEBO-GII AES implementation, the FPGA uses significantly less power than average when a byte being written into the round register has the value 0. It uses even less power when a byte written into the round register is the same value as the overwritten byte. Knowledge of these effects, often discovered through trial-and-error, is helpful in formulating selection functions.

In many cases, the secret constant targeted by a DPA attack is a recognizable portion of a round key. In other cases, such as when analyzing DES-X, HMAC-SHA [35], or AES in counter-mode with unknown starting counter [27], the attack recovers a constant that is a function of multiple secret constants (or keys). An attack may need to be iterated for a number of rounds to recover enough secret materials to solve for all keys. In some cases (such as HMAC-SHA), instead of recovering the original key, the analysis yields intermediate values which enable the attacker to perform the same operations as with the original key.

In some instances, known key analysis may show fairly large leaks, but targeting the device specific leakages may appear to require guessing a large number of bits of the key or sensitive parameter. In cases where an attacker can select the cipher input, the analysis complexity can often be reduced, allowing a practical attack against the leak. With the cipher MISTY, e.g., chosen messages can reduce an attack from 32-bits to 16 [28]. Similarly, with AES, an attack on register contents after the MixColumns step can be reduced from 32-bits to 24 or 16 using chosen messages [26].

Chosen message analysis is particularly helpful with public key algorithms. One strategy frequently applied with public key algorithms is to guess only part of the key (e.g., a few most significant bits) and predict intermediates using this value. The best prediction will provide the closest approximate of the key, yet still deviate from the actual intermediate state (see Sect. 4.4.1). This approach can be applied iteratively, to successively obtain better approximations until the key is known.

Selection functions are normally 0/1 valued. In some cases, especially when leakage characteristics of a device are well known, a selection function can have a non-binary

output. This output is then used as a weight in the final averaging step, where the weights can be zero, positive or negative. This type of analysis is closely related to Correlation Power Analysis described in Sect. 5.1. In general, non-binary and multi-bit selection functions can improve the efficiency of attacks when assumptions about device leakage hold, whereas binary, single-bit selection functions can be useful without requiring additional assumptions about multi-bit leakages.

A final step of the hypothesis generation phase is to compute the output of each selection function for each trace. For example, if 256 selection functions are being tried to solve for a byte of an AES key, the input to this phase would typically be the ciphertext or plaintext for each trace, and the output would be a vector of 256 bits for each trace.

4.3 DPA: averaging

The analysis stage is where the core DPA calculations are performed. In addition to computing the averages of various subsets of the traces, the average of all traces and the variance at each point across all traces are also generally computed.

Averaging performance is determined by processing power and storage throughput. A number of performance optimizations are helpful when working with large data sets. The basic task is to rapidly compute averages of many subsets of traces. A large analysis might involve 10^8 traces, each with at least a few hundred points, with 10^5 subsets to average together. In terms of number of traces (N), length of traces (L), and number of selection functions (M), the naive complexity of this task is $O(N \cdot M \cdot L)$ and the memory use is $O(M \cdot L)$.

Optimization can simplify the problem. DPA involves comparing the average of the subset where a selection function is 1 (A_1) to the average of the subset where the selection function is 0 (A_0). If the average of all traces (A_{all}) is known, then A_0 can be calculated from A_1 and A_{all} using the number of traces in A_1 and the total number of traces. Thus, when calculating many selection functions over the same set of traces, it is generally sufficient to calculate A_{all} and the $A_{1,m}$ subsets since $A_0 \approx A_{all} - A_1$. This results in just a factor of two improvements, so the complexity is still $O(N \cdot M \cdot L)$, but CPU and memory use is halved.

A second optimization when calculating many selection functions over the same set of traces is to use a cache. A cache size of $2^8 - 1$ works with 8 traces at a time, computing the sums of 255 possible combinations of these traces. Then, these traces can be added into the mth averaging task using one addition out of the cache instead of up to 8 additions of the individual traces. A cache of size c requires $O(c \cdot L)$ memory and takes $O(2^c \cdot L)$ operations to set up. Performance using a cache is improved to $O\left(\frac{N}{c} \cdot M \cdot L + 2^c\right)$.

Many analyses have statistical bottlenecks. The attack on AES in Sect. 2.3 predicted the output of SubBytes in the first round. Guessing one byte of the round key allows 8 bits of intermediate data to be predicted. If DPA tests are performed for each of these bits, for each value the key byte could take, then 2, 048 selection functions will be evaluated. Because the prediction about the intermediate depends on only one byte of the ciphertext, only 256 unique sequences will be observed over the set of L traces. In general, if there is a statistical bottleneck in the selection function outputs, and only B unique sequences of selection functions are generated over our L traces (and $B << L$), then using an "input bins" cache with B entries can improve performance to $O\left(L + \frac{B}{c} \cdot M \cdot N + 2^c\right)$.[12] Bottlenecks can be introduced by repeating each message multiple times.

As a result, any decrease in the trace size produces a proportional improvement in averaging time. For example, trace regions before or after the cryptographic operation can be deleted. Compression methods (as discussed in the signal processing subsection above) can also greatly reduce the number of points that need to be averaged.

When evaluating extremely high numbers of selection functions (e.g., 2^{32}) the selection function generation and evaluation stages can be folded into the averaging step [36].

Although DPA tests normally compare the difference of averages, this test is not effective in all cases, especially in the presence of countermeasures, and in some cases the signals require preprocessing before averaging (see Sect. 5.2). In other cases, distributions such as those shown in Fig. 5 may also be analyzed. In this case, instead if computing the average of the subset traces at each point, this step would be modified totally the frequency distribution of samples at each point.

The task itself is also embarrassingly parallelizable. Data can be distributed over many drives to eliminate I/O bottlenecks. Computation can be distributed over multiple threads or machines. Although few devices today require such large data sets, optimizations can make working with billions or potentially even trillions of traces practical [36].

4.4 DPA: evaluation

In simple cases, the results of DPA can be evaluated using visual inspection by a human operator. The correct key guess results in large peaks in the differential trace, while much smaller peaks are visible for incorrect key guesses. It is also easy to develop automated tools to measure peaks and list

[12] Alternatively, if the cache optimization is not used, a bottleneck still can be exploited in a very memory-efficient approach. After the B averages are computed, the average for each of the m bins can be computed one after the other and saved to disk. This approach involves only $O((B+2) \cdot L)$ memory and $O(L + (B \cdot M \cdot L))$ complexity.

or plot their amplitudes, or even suggest the most likely key guesses.

Regions with unusually high noise can show spurious spikes in a differential trace. To correct for these effects, and to help assess the statistical significance of the results, each point in the differential trace can be divided the standard deviation of all traces at that point. The result is a normalized trace giving the polarity and significance of the difference (measured in standard deviations) at each point in time.

The "difference of averages" is just one way that two distributions of measurements can differ. A more general statistical test can compare at the distribution of measurements at each point in the subsets of traces, and calculates the significance of differences observed between them. For such analysis, the "averaging" stage actually computes the distributions of measurements rather than compressing the distributions down into their averages.

For some algorithms, and for certain types of DPA attacks, the evaluation process is more complex, since there are be multiple other guesses besides the correct key which have significant correlation to the target leak and thus may show spikes in the differential trace. These guesses are termed as "harmonics" of the correct key and these will be described in the next subsection.

For some algorithms the attack is applied iteratively, as new information about the key enables the generation of new selection functions. In these cases, the DPA evaluation stage is not the final stage of an attack. The iteration process usually restarts back at the selection function generation stage, but for adaptive chosen message attacks, the evaluation result guides the next sets of inputs for the data collection stage. Iteration is also required for algorithms such as AES-256 and triple DES where multiple round subkeys or multiple encryption keys must be found.

4.4.1 Harmonics

The AES selection function described in Sect. 2.3 targets a bit of SubBytes output. When the guessed byte is correct, its output is correct (i.e., equals the target bit for each input message) with probability 1. The assumption that when the guessed byte is incorrect the output from the prediction is uncorrelated (i.e., can be treated as random) is an approximation; smaller positive or negative correlations exist.

These smaller correlations are ignored in a basic DPA analysis, but can provide additional information for identifying the correct key guess. A relatively extreme example of this occurs in S-box 2 of the DES standard (Fig. 17) in output bit 2 (where 0 is the least-significant bit). If the guessed value for the corresponding six key bits equals the correct value except that its high order bit is complimented, then output bit 2 for this S-box output will be incorrect in 60 out of 64 cases. As a result, when predictions for output bit 2 of S-box 2

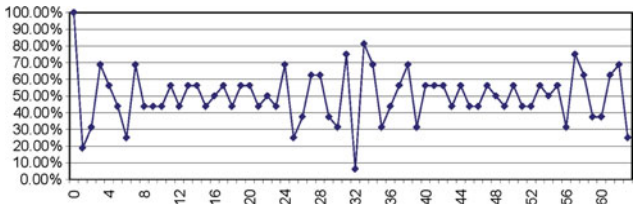

Fig. 17 Biases observed for incorrect key guesses of DES S-box 2, output bit 2. The *X* axis shows the XOR difference between the tested 6 key bits and the correct key bits. The *Y* axis shows the likelihood the incorrect key will correctly predict the output bit

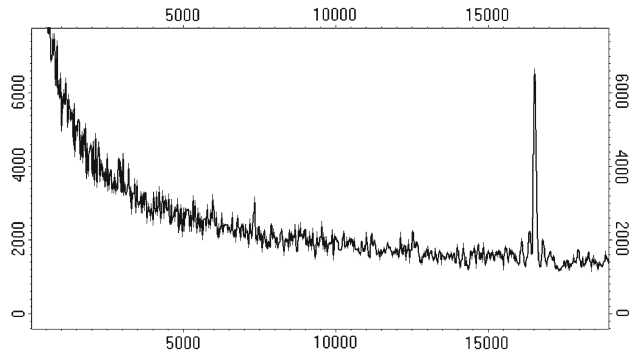

Fig. 18 Graph of signal amplitude observed for each key guess in an attack on RSA-CRT reduction-by-division, from 0x8000 to 0xC9FC

are used as a selection function, the correct value will appear along with a harmonic peak[13] of opposite polarity. Although this high-intensity "harmonic" peak could initially be misinterpreted as the correct one, the *pattern* of the peaks can actually help an adversary by providing additional a way to confirm key guesses.

Harmonics also often appear in public-key cryptography. Two examples are the Hamming weight pattern observed in the MRED attack on RSA-CRT [25], and the patterns often observed in attacks based on the multiplication or division. Figure 18 shows an example of such harmonic patterns, as seen in an attack targeting intermediates in the RSA-CRT initial reduction step.

In Fig. 18, the target device implements the RSA-CRT initial reduction step by finding the remainder of the long division of ciphertexts C by a secret prime p. The X axis indicates the hypothesis tested, with the left edge of the screen corresponding to hypothesis hexadecimal `8000` and the right edge corresponding to hypothesis hexadecimal `C9FC`. Selection functions were generated by predicting the 8th most-significant bit of the quotient of each trace's ciphertext C divided by the 15-bit hypothesis value. The Y axis at each offset plots the amplitude of the strongest signal observed in

[13] DPA literature refers to harmonics as either "ghost" peaks or "spurious" peaks. We chose to use the more positive term, "harmonics", to describe these peaks, since we believe harmonics actually help the DPA process.

the differential trace produced using the hypothesis. A strong spike peaking in the range C074-C092 is visible, with side lobes in neighboring X values.[14] For guesses near 8000, the selection function is correlated with RSA-CRT input bits, causing elevated signals at the left side of the figure.

In algebraic relationships, hypotheses that are off by $\pm\epsilon$ are often strongly correlated to the correct value when ϵ is small. For slightly larger ϵ, the correlation may pass through zero, turn negative, and then become positive again. This is seen in Fig. 18 and is due to the fact that in division when the 15-bit hypothesis is correct in the highest 8 bits, the predicted quotient will also be roughly correct in the highest 8 bits (but may be off by a small amount due to carries from the lower-order bits).

As ϵ increases, so do systematic errors in the predicted value. When the guess of p is slightly larger than the correct value, the predicted quotient will be systematically underestimated and vice versa. For very small ϵ, this systematic error will be close to zero. For slightly larger ϵ, the prediction is systematically biased towards being off by 1 or -1, causing the DPA spikes to be anticorrelated. For even larger ϵ, the prediction will be off by ± 2, causing DPA spikes to be correlated again. As ϵ increases, the degree of correlation decreases due to other carries accruing in the calculation. While the direction of the correlation alternates, the amplitude decreases, causing the characteristic "wavelet" pattern observed in Fig. 18.

4.4.2 Dealing with harmonics

Although harmonics can usually be ignored, they can provide useful information. The pattern of harmonic peaks can help identify the correct value. Harmonics are a function of the target algorithm and the selection function strategy employed. It is, therefore, possible to analyze the selection function process, determine the pattern of harmonics it would generate, and then match this pattern against the amplitudes of the observed harmonics.

In cases such as DES when the harmonic patterns vary from bit to bit, harmonics from each S-box output bit can provide additional confidence in the correct result. For each bit, the fundamental signal will correspond to the correct key guess, but the pattern of harmonics will vary. Alternate multi-bit evaluation strategies such as correlation power analysis (discussed in Sect. 5.1) are also less sensitive to harmonic peaks.

Harmonics such as those observed in Fig. 18 may blur the amount of low-order-bit information one can recover from a typical attack. In this case, the attack can be repeated while targeting successively narrower ranges of values for p. Each iteration of the attack recovers lower-order bits of the quotient, ultimately permitting all of p to be recovered.

[14] Note that this observed peak does indeed correspond to the correct value of the key, which begins with the sequence C083A6.

5 Variants of DPA

In addition to SPA and DPA, there are variants of the basic attack that are better suited to exploit information leakage in some settings. We briefly describe some of these techniques here.

5.1 Correlation power analysis

Correlation power analysis (CPA) [37] involves evaluating the degree of correlation between variations within the set of measurements and a model of device leakage that depends on the value of (or a function of) one or more intermediates in the cryptographic calculation. Common examples include correlating power measurements with the Hamming weight of a multi-bit value in a register or on a bus or the Hamming distance between a value and the value it overwrites.

Correlation power analysis is most effective in white-box analysis where the device leakage model is known. It can also be used for black-box evaluations as long as there is some correlation between the actual leakages of the device and the leakage model being used for CPA.

If the number of traces available is limited, CPA can help make the maximum use of the data. If CPA does not find leakage, it is ambiguous whether this indicates low leakage in the device or a large gap between actual leakage and the model. Similarly, if CPA is successful at recovering the key, additional leakage modes may also be present.

In our experience, Hamming weight or Hamming distance models can be reasonable approximation for leaks from software running on some 8-bit microcontrollers. For larger ASICs and CPUs, these methods are less effective at modeling the leakage found. Because these models do not exactly reflect a device's properties, countermeasures developed solely on the basis of these simplistic models are generally insufficient to prevent DPA attacks.

5.2 Probability distribution analysis

The subset of data from applying a correct selection function may have a different probability distribution from the overall data set, yet have a mean that is statistically indistinguishable. For example, consider the case where the points at a given offset in the subset of traces selected by a selection function have the same mean but a smaller standard deviation than the overall data set (see Fig. 19). Such a distribution could arise, e.g., in a hardware implementation that uses the masking countermeasure, but processes the masked data and the mask concurrently (see Sect. 6.2). A standard DPA

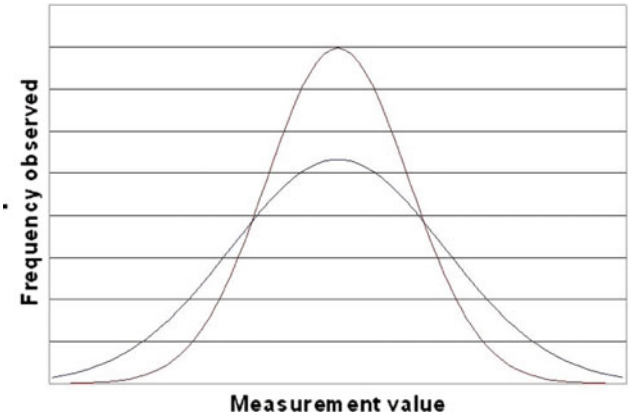

Fig. 19 Example where a binary selection function creates two distributions with the same mean but different standard deviations

performed at this location will not have any peaks, yet the target device can be broken by adapting the attack to distinguish the distributions. For the distribution in Fig. 19, a straightforward approach is to preprocess traces by first subtracting from each trace the average of all traces collected, then squaring the value of each remaining data point. The DPA analysis performed on the preprocessed traces is equivalent to comparing the variances of the data at each point, rather than their means. This process actually performs the zero-offset, second-order DPA attack (ZO2DPA) described in [34].

More generally, given any differing probability distribution, it is possible to define a trace preprocessing operation that will make DPA work. Alternatively, the DPA analysis can be performed by comparing probability distributions instead of averages. Techniques such as Mutual Information Analysis [38] are also based on this principle.

5.3 DEMA and other side channels

Both the basic DPA test described in Sect. 2.2 and the SPA techniques from Sect. 3 work equally well on electromagnetic (EM) measurements. SPA and DPA techniques applied to EM measurements are termed simple electromagnetic attacks (SEMA) and differential electromagnetic attacks (DEMA) in [39–41]. Such attacks can be highly effective, particularly if power measurements are unavailable.

Data from other sources could potentially be used in DPA-style analysis. Potential side channels such as photon emissions from semiconductors [23] and temperature measurements are areas for further research. To extract keys from obfuscated cipher implementations, digital data dumped from repeated cryptographic operations can be used in lieu of power traces. Cache timing and other microarchitectural attacks [42–44] also use similar statistical techniques, but make somewhat different assumptions about the type of access available to an attacker in order to break software implementations of cryptography.

5.4 High-Order DPA

The DPA paper [11] introduced High-Order DPA, saying that "Of particular importance are high-order DPA functions that combine multiple samples from within a trace". High-Order DPA is an analysis method that targets a known or hypothesized relationship between parameters contributing to a side channel.[15] The "order" of a high-order attack is the number of parameters involved in the target relationship. High-order DPA can help analyze relationships such as:

Similarity/difference The calculations at different points in a power trace (or in a pair of traces) may involve a common data parameter. High-order combination functions that measure correlation or covariance can be used to detect these relationships. We have successfully used this approach in several cross-correlation attacks on modular exponentiation and ECC.[16]

Masked shares of a secret This is the traditional second-order DPA attack described in literature [45,46,34,47–50]. For example, in a masked implementation, a sensitive intermediate may be manipulated as two parts. Each part by itself is random, but the exclusive-or of the parts would give the intermediate. Individual measurements are correlated to the parts, but uncorrelated to the variable. A high-order function can combine a measurement correlated to the first part with a measurement correlated to the second part, so that the combination is correlated to the sensitive variable. For example, if two points are correlated to A and B respectively, and the secret intermediate is $A \oplus B$, the product of points A and B will show correlation to $A \oplus B$ and can be used to test hypotheses about the secret.[17] Filtering attacks are another simple high-order method for targeting leaks such as this. For this attack, a standard DPA analysis is performed using only the traces with below-average measurements at the first point. Within these traces, A is biased, so steps that manipulate B will be correlated to $A \oplus B$.

[15] Unlike trace compression methods that simply combine multiple measurements in order to shrink traces, a high-order analysis targets distinct parameters involved in the target device's computation.

[16] For example, in binary left-to-right modular exponentiations, squaring steps use only the previous result, while multiplication steps also involve the input base. The correlation between two nonadjacent steps will be higher if both operations are multiplications by the base. The "doubling attack" [29] is another application of this method, using chosen messages.

[17] An even better high-order function is $(A - < A >) \cdot (B - < B >)$ where $< X >$ is the average of all traces at point X.

Unknown input Measurements at a first location in a set of power traces can be used to estimate the value of otherwise unknown inputs and outputs to cryptographic functions. These estimates can then be used to prepare probabilistic predictions for plaintext/ciphertext portions needed for a standard DPA attack which is then evaluated at a second location in the set of power traces.

Note that when the exact relationship between data values in a calculation is known, high order attacks can, in some cases, be avoided. For example, DPA can break AES in counter mode with unknown initial counter using a first order attack [27]. This attack takes advantage of the fact that, although the input is unknown, it can be expressed as the sum of a known value with a secret constant. The secret constant can thus be treated as part of the key in a modified first-order DPA analysis. The MRED attack on RSA [25] is also a variant of this approach.

5.5 Template attacks

Template attacks [51,52] seek to make maximal use of a small number of traces from a target device. In a template attack, the analyst constructs a model of the target device. In contrast to CPA, template attacks build a model from actual power measurements or simulations. This model is typically represented by a set of statistical parameters for the expected power traces corresponding to various states that the device may enter. Once these templates have been created, they can be used to determine the most likely state of a target device from a small number of actual traces.

In some cases, the model (set of templates) may be constructed using the actual device that is the target of the attack. For example, programmable devices such as smartphones often provide untrusted code with access to the same cryptographic primitives used with high-value keys. In other cases, the templates may be built using a device from the same family as the target. Still other template attacks may use templates approximated from simulations of the target device.

Although template attacks may require a large amount of initial effort, they can achieve theoretically optimal use of the signal in the target traces. In situations where parameterized models for leakage from a target device family already exist, the stochastic approach [53] can be a much more efficient alternative to template attacks. If the leakage models are accurate enough, the effectiveness of stochastic methods can approach that of template attacks.

5.6 Reverse engineering unknown S-boxes and algorithms

Custom ciphers are sometimes encountered in black box evaluation of cryptographic devices. In cases where the custom cipher is based on a well-known design, but with different set of constants such as S-boxes, etc., DPA can be used to first reverse engineer the values of these customized constants and then attack the cipher. In cases where the design itself is secret, reverse engineering of the design solely using SPA and DPA can be a challenge but has been done in some cases [54,55]. A general approach is to iteratively evaluate a broad range of selection functions. Those yielding correlations that are stronger or later in time may correspond to the device's intermediates deeper in the algorithm. The better selection functions are kept and their outputs are used as candidate inputs for subsequent iterations of the analysis.

6 Preventing DPA

A first step in preventing power analysis is to eliminate large leaks that can create SPA vulnerabilities. In particular, implementations should use constant execution paths and avoid taking conditional branches on secret data. Where possible, processing primitives and instructions should be selected from those known to leak less information in their power consumption.

Preventing DPA requires additional effort. DPA can exploit very small rates of information leakage, since the statistical test accumulates signals correlated to a target secret and diminishes the effect of noise and uncorrelated device activity. Techniques for preventing DPA are described in the following sections.

6.1 Leakage reduction

Some approaches make DPA more challenging by decreasing the *signal-to-noise* ratio of the power side channel—either by decreasing leakage (signal) or increasing noise. A decrease in the signal-to-noise ratio will increase the number of traces required for a successful attack.

6.1.1 Balancing

Leakage may be reduced by structuring cryptographic circuits to use a more balanced amount of power. Balancing aims to reduce signal by making the amount of power used less dependent on data values and/or operation type (Fig. 20).

Balanced gate constructions using multi-bit data representations and balanced transitions were first described in [11,56,57]. Subsequently, there has been much research on developing and implementing DPA-resistant logic styles. Several works have focused on dual-rail precharge logic [58–63]. Other proposals have been based on current mode [64,65] and asynchronous logic styles [66,67].[18] While it is

[18] Chapter 7.3 in [10] is a good reference on this topic.

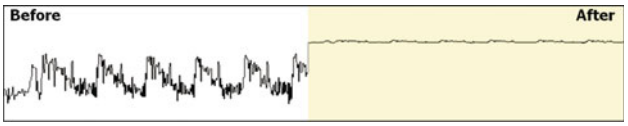

Fig. 20 Balancing power consumption reduces the observed amplitude of data dependant variations (simulated data)

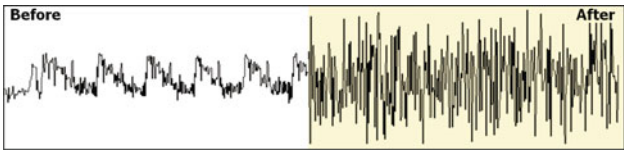

Fig. 21 Example of adding amplitude noise to obscure and reduce signal (simulated data)

possible to implement cryptographic cores using logic styles that minimize data-dependent leakage, it is a challenge to do so effectively in modern ASICs using standard tools and design flows. The power consumption of modern ASIC components depends on wire routing, gate output driver strengths, capacitive effects, cross-talk between wires, and other properties that are difficult to control at design time or that may be changed by downstream tools.

At a coarser level, compensating circuitry and physical shielding can also be applied to regulate and balance the power consumption at the circuit or chip level [68,69].

6.1.2 Amplitude and temporal noise

A second approach involves introducing noise into the measurements [70].

Amplitude noise is added by adding circuits that consume a variable amount of power, or using circuit elements to perform calculations that are uncorrelated to the cryptographic intermediates being hidden.[19] With amplitude noise, only the spectral component of noise that is frequency-matched to the signal is relevant (Fig. 21).

Temporal noise is introduced by inserting variations in timing and execution order. Methods include using deliberately decorrelated and varying clocks, random wait states, random execution re-ordering, use of dummy operations, and random branching [70]. These techniques reduce an adversary's ability to guess when a specific sensitive operation has occurred (Fig. 22).

6.1.3 Effectiveness

A factor of k reduction in the signal-to-noise ratio using leakage reduction techniques such as balancing, increases the number of traces required for DPA by a factor of k^2. However, there are practical limits to how well this can be done in modern ASIC design.

[19] If "noise" is positively correlated to intermediates, the attack gets easier. If noise is negatively correlated to intermediates, the result is a balancing countermeasure.

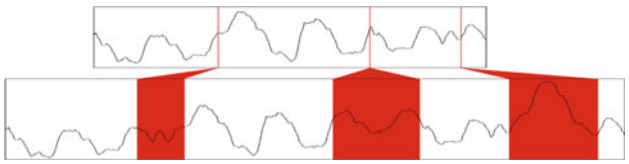

Fig. 22 Delays inserted into a power trace sequence make alignment and synchronization more difficult (simulated data)

The effectiveness of a single amplitude or temporal noise introduction countermeasure is reduced if signal processing techniques can detect and partially (or fully) eliminate the additional noise. For example, filtering can eliminate amplitude noise that is not frequency matched to the signal. Similarly, dummy operations may have a distinct power profile that can be detected (whether reliably or probabilistically) and removed from a power trace. Combining multiple amplitude and temporal noise introduction countermeasures and leakage reduction techniques usually makes noise reduction by signal processing much harder. Furthermore, such a combination further increases the difficulty of DPA attack workflow steps that use automated (and frequently independent) mechanisms to improve trace alignment, synchronize common trace features, and reject noise.

To the extent that amplitude and temporal noise are not removable by signal processing, these countermeasures increase the number of traces required by DPA. A factor of k reduction in the signal-to-noise ratio achieved by amplitude noise increases the number of traces required by a factor of k^2. Temporal noise is less effective; a reduction in the probability of predicting when a critical operation occurs by a factor of k only increases the number of traces required by a factor of $O(k)$. This is because, in this case, each individual trace provides $O(k)$ more locations to perform the attack. (See [33] for additional details on why temporal noise is less effective.)

While countermeasures targeting the signal-to-noise ratio cannot reduce information leakage to zero, they can substantially increase the number of samples required by an attacker by reducing the information content of each trace. Bounding the amount of information leaked from individual traces is also a pre-requisite for other countermeasures described later.

6.2 Incorporating randomness: blinding and masking

Countermeasures based on masking or blinding resist DPA by randomly changing the representation of secret parameters. By periodically updating the representation, the number of traces that use any one representation of the secret is limited. This disrupts statistical tests such as first-order

DPA from accumulating information about computational intermediates that are directly related to a target secret.

6.2.1 Blinding public key algorithms

Many public key cryptographic systems involve mathematical computations over a finite field that can be masked or blinded in a variety of ways. For example, the following equations show several relationships that hold in modular arithmetic that enable secret values to be randomized without changing the ultimate result:

$$A^{d+k\cdot\phi(P)} \bmod P = A^d \bmod P$$
$$((A \bmod (kP)) \bmod P) = (A \bmod P)$$
$$A^d \bmod P = \left(\left(A^r\right)\cdot\left(A^{d-r}\right)\right) \bmod P$$
$$A^d \bmod P = \left(\left(A^r\right)^{d\cdot r^{-1} \bmod \phi(P)}\right) \bmod P$$
$$\text{and,}\quad \text{for } ((d\cdot e) == 1 \bmod \phi(N)) :$$
$$A^d \bmod N = \left(\left(A\cdot B^e \bmod N\right)^d \bmod N\right)\cdot B^{-1} \bmod N$$

Similar relationships hold in many other finite fields used in cryptography.

In a blinded implementation [5,71,72], secret parameters are masked by combining each with one or more random blinding factors. The blinding factor(s) and blinded secret together comprise a blinded representation of that parameter. The cryptographic algorithm is then implemented using the blinded representation. Blinding factors and blinded secrets are updated frequently between, or even during, computations. The update process mixes information unknown to the attacker into the blinded representation so that partial information leaked about past blinded representations is difficult to combine with partial information that may leak about the updated representation.

6.2.2 Masking symmetric algorithms

Symmetric algorithms can also be masked. Secret constants and intermediates can be split into multiple randomized parts. For example, in Boolean masking, intermediate data byte X is masked by generating a random byte R and representing X by the pair (A, R) where $A = X \oplus R$ [45,73,74]. Additive masking works similarly, representing the byte X by the pair (A, R) where $A = X + R \bmod 256$. Other masking operations can also be used.

A masked cipher implementation stores all key bits and round intermediates in masked representations. Computations are performed on independent parts which are not reconstituted until the final output from the final round is complete.

Symmetric algorithms include nonlinear operations, such as substitution tables, that must be adapted to work with masked inputs and outputs. One approach involves transforming the nonlinear operation to accept masked inputs and produce masked outputs. For example, if a cipher uses a substitution table S and will apply a Boolean mask R_0 to the input and requires that the output be masked with R_1, the substitution table S' where $S'[i] = S[i \oplus R_0] \oplus R_1$.

Other approaches involve decomposing a complex nonlinear operation into a sequence of linear and nonlinear operations, where the nonlinear operations are simple enough to be adapted to work with masked inputs and outputs [75–77].

Masked representations need to be updated frequently with fresh randomness. Mask changes help prevent DPA attackers from accumulating information about (or solving for) the masking parameters.

The goal of blinding and masking is to ensure that information leakage is not directly related to the secret data. Because the individual values of blinding factors, masks, and blinded or masked secrets are not correlated to the secret data, leaks from only one of these values cannot be used to recover the secret.

This strategy forces attackers to shift to a high-order DPA attack targeting the relationship between blinding factors and blinded secrets, or between masks and masked data. The number of traces required for high-order analysis can be much higher than for DPA. For example, in low signal-to-noise environments and for certain kinds of leakages [45] shows that the number of samples for a successful high-order attack grows exponentially in the number of parts that the secrets are split.

6.3 Protocol level countermeasures

The most effective and least difficult way to address side channel attacks is to design cryptographic protocols to survive leakage. There is usually no way that a designer can ensure that, across all operations an adversary could observe, the leaked information will not approach the size of the key. Evaluation processes also cannot ensure the absence of minuscule side channels, even if extensive balancing and noise generation countermeasures are attempted. For comparison, devices implementing conventional protocols can be straightforward to break with leakage rates of 10^{-9} bits per operation or less, whereas protocol-level countermeasures can preserve security with leakage rates exceeding 10 bits per operation—a difference of more than ten orders of magnitude. Thus, cryptographic protocols that are resilient to leakage are crucial for implementations using less-than-perfect hardware.

A general approach for surviving leakage is to limit the number of transactions that can be performed with any given key. For example, consider the case of a device with a 256-bit key that can be operated at most ten times, and which will

destroy its keys after the 10th transaction attempt.[20] If the desired security level is 192 bits, then this design can tolerate a cumulative leak of 64 bits over its lifetime, which is achieved if the average leak is <6.4 bits per transaction.[21] Subsequent constructions in this section will also use this concept of a maximum net leakage that the design can tolerate and a corresponding maximum leakage rate (L_{MAX}) per transaction.

Key update procedures enable leak resilience in devices to support larger numbers of transactions. As with masking and blinding approaches, a key update procedure is performed at periodic intervals. The update frequency is chosen so that L_{MAX} times the number of uses for any one key value does not cross the design's security threshold.

The purpose of a key update transform is to mix the key state such that incomplete information below the desired security threshold about the pre-transformed key cannot be usefully correlated to the post-transformed key. If done properly, the update process will make it cryptographically hard for an adversary to accumulate useful information across transforms. For example, if a 256-bit key is hashed (e.g., with SHA-256) between transactions to produce a new 256-bit key, then a security level of 192 bits can be preserved provided that the total information leaked in deriving each temporary key, using the key in a transaction, and deriving the next key totals <64 bits (i.e., $L_{\mathrm{MAX}} = 64/3$).

Key updates and key derivation may be structured in a hierarchically defined key-tree, enabling a server in a multi-party protocol to efficiently derive the current key derived by a tamper resistant client device. A key-tree is a tree structure defined from a root secret key and a set of key update transforms. For a typical binary key tree, two transforms are defined, enabling two child nodes to be derived from each parent node. Lowest-level nodes can be derived quickly from the root or, if the update operations are invertible, from other nodes. Counters or other protocol constructions limit the number of times any given node is used to form transaction keys. Key trees were first described in [78,79], and the approach is now used in many systems including the EMV payment protocol [80].

The key-tree construction can be extended to create and validate symmetric message authentication codes (MACs) in a leakage-resilient manner. The tree's root is the MAC key, and the message being authenticated (or its hash) defines the path to a leaf node that determines the MAC value. Leakage-resilient MACs, key derivation, and nonlinear key updates can be combined to perform DPA-resilient authenticated bulk encryption/decryption [81]. In this construction, an encryption session key is derived from a shared key using a key-tree, and the hash of the ciphertext is authenticated using a key-tree based MAC. Bulk encryption is performed using a sequence of encryption keys derived from the initial encryption session key using a non-linear update function, limiting the number of message blocks for each encryption key. The recipient verifies the MAC of the ciphertext hash before decryption, thus preventing any DPA attack based on decrypting multiple altered ciphertexts with the same key. Unlike other constructions, this approach is effective for applications such as storage encryption which require repeated decryption of the same ciphertext. In this case, the construction's security requires a bound on the implementation's leakage when repeatedly operating with the same inputs (message and key).

In recent years, the topic of leakage-resilient cryptography has received much attention within the scientific community [82–90]. The focus of such research has been the construction of cryptographic primitives whose implementations can be proven to be secure under certain leakage models. While more work is required to make the leakage models used in some of the proofs more realistic, both leakage models and leakage-resistant constructions are important areas of research. Work to integrate leakage resilience into protocols is also needed, and can greatly improve hardware security by aligning protocols' security requirements with the properties of actual devices.

7 Conclusion

Modern cryptographic primitives such as AES, RSA, ECC, HMAC, etc., are designed to resist attacks by adversaries with access to plaintext and ciphertext data. A cryptanalytic result that yielded a small improvement over brute force would be considered a major breakthrough, even if the attack required resources far beyond the reach of normal adversaries.

DPA can accomplish in minutes or days what decades of cryptanalytic work cannot: the extraction of secret keys from devices using completely correct implementations of strong primitives. Even if the amount of information in each trace is orders of magnitude below the resolution of the measurement apparatus, this additional information can convert the computationally infeasible problem of breaking a cipher using brute force into a computation that can be performed quickly on a PC.

When we first discovered DPA, we realized that the technique would have a major impact on cryptosystem implementations. Nevertheless, we were surprised by how quickly researchers and practitioners directed attention to the area,

[20] Failure counters are necessary to prevent adversaries from exceeding the transaction limit by interrupting power or resetting the device after the cryptographic operation but before the counter update. Counters should also be updated prior to the transaction.

[21] For most devices, any transaction limits that could be applied would result in L_MAX being too low to be of much use. For example, if the device could perform 2^16 operations, that maximum tolerable leakage per operation would be $\approx 10^{-3}$, which is too small to be realistic or a verifiable design threshold.

and we are amazed by the tremendous body of research that has emerged on this topic over the last decade. At a broader level, this body of new work has helped reinvigorate research on the practical problems of understanding how actual cryptosystems fail and how they can be protected. While modern algorithms provide extraordinary resistance against conventional cryptanalysis, a much more research is needed to help implementations achieve equivalent trustworthiness and strength against DPA and other attacks. We appreciate the efforts of all the researchers who have built upon and extended our work to create the field of side-channel cryptanalysis, and who are helping to train the next generation of engineers so that future cryptographic devices will be better defended than those of the past.

Acknowledgments The authors would like to thank their colleagues Jeremy Cooper, Gilbert Goodwill, Chris Gori, Nate Lawson, Mark Marson, Trevor Perrin, Takeshi Sugawara and Luke Teyssier for their contributions to the development of the DPA Workstation platform and to its analysis and visualization tools that were used in this paper. We would also like to thank the RCIS team at AIST Japan and Tohoku University for creating and generously sharing the SASEBO platform which was used in this paper.

References

1. Biham, E., Shamir, A.: Differential Cryptanalysis of DES-like Cryptosystems. In: Menezes, A, Vanstone, S. A. (eds.) CRYPTO, Lecture Notes in Computer Science, vol. 537, pp. 2–21. Springer, Berlin (1990)
2. Matsui, M., Yamagishi, A.: A new method for known plaintext attack of FEAL Cipher. In: EUROCRYPT, pp. 81–91 (1992)
3. Boneh, D., DeMillo, R. A., Lipton, R.J.: On the importance of checking cryptographic protocols for faults (extended abstract). In: EUROCRYPT, pp. 37–51 (1997)
4. Biham, E., Shamir, A.: Differential fault analysis of secret key cryptosystems. In: Kaliski, B.S. Jr. (ed.) CRYPTO, Lecture Notes in Computer Science, vol. 1294, pp. 513–525. Springer, Berlin (1997)
5. Kocher, P.C.: Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In: Koblitz, N. (ed.) CRYPTO, Lecture Notes in Computer Science, vol. 1109, pp. 104–113. Springer, Berlin (1996)
6. Dhem, J.-F., Koeune, F., Leroux, P.-A., Mestré, P., Quisquater, J.-J., Willems, J.-L.: A practical implementation of the timing attack. In: Quisquater, J.-J., Schneier, B. (eds.) CARDIS, Lecture Notes in Computer Science, vol. 1820, pp. 167–182. Springer, Berlin (1998)
7. Anderson, R., Kuhn, M.: Tamper resistance—a cautionary note. Second Usenix Workshop on Smartcard Technology **1**, 1 (1996)
8. Anderson, R.J., Kuhn, M.G.: Low cost attacks on tamper resistant devices. In: Christianson, B., Crispo, B. Lomas, T.M.A., Roe, M. (eds) Security Protocols Workshop, Lecture Notes in Computer Science, vol. 1361, pp. 125–136, Springer, Berlin (1997)
9. National Security Agency. NACSIM 5000 TEMPEST FUNDAMENTALS. http://cryptome.org/jya/nacsim-5000/nacsim-5000.htm (1982)
10. Mangard, S., Oswald, E., Popp, T.: Power analysis attacks: revealing the secrets of smart cards. Springer, New York (2007). ISBN: 978-0-387-30857-9
11. Kocher, P.C., Jaffe, J., Jun B.: Differential power analysis. In: Wiener, M.J. (ed.): Advances in Cryptology—CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15–19, 1999, Proceedings, Lecture Notes in Computer Science, vol. 1666, pp. 388–397. Springer, Berlin (1999)
12. Novak, R.: SPA-based adaptive chosen-ciphertext attack on RSA Implementation. In: Naccache, D., Paillier, P. (eds.) Public Key Cryptography, Lecture Notes in Computer Science, vol. 2274, pp. 252–262. Springer, Berlin (2002)
13. Schramm, K., Wollinger, T.J., Paar, C.: A new class of collision attacks and its application to DES. In: Johansson, T. (ed.) FSE, Lecture Notes in Computer Science, vol. 2887, pp. 206–222. Springer, Berlin (2003)
14. Ledig, H., Muller, F., Valette, F.: Enhancing collision attacks. In: Joye, M., Quisquater, J.-J. (eds.): Cryptographic Hardware and Embedded Systems—CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11–13, 2004. Proceedings, Lecture Notes in Computer Science, vol. 3156, pp. 176–190 Springer, Berlin (2004)
15. Daniel Bleichenbacher. Bell Laboratories. Private Communication to authors
16. Nguyen, P.Q., Shparlinski, I.: The insecurity of the digital signature algorithm with partially known nonces. J. Cryptol. **15**(3), 151–176 (2002)
17. Howgrave-Graham, N., Smart Nigel, P.: Lattice attacks on digital signature schemes. Des. Codes Cryptogr. **23**(3), 283–290 (2001)
18. Boneh, D., Shparlinski, I.: On the unpredictability of bits of the elliptic curve diffie–Hellman scheme. In: Kilian, J. (ed.) CRYPTO, Lecture Notes in Computer Science, vol. 2139, pp. 201–212. Springer, Berlin (2001)
19. Mangard, S.: A simple power-analysis (SPA) attack on implementations of the AES key expansion. In: Lee, P.J., Lim, C.H. (eds.) ICISC, Lecture Notes in Computer Science, vol. 2587, pp. 343–358. Springer, Berlin (2002)
20. Renauld, M., Standaert, F.-X., Veyrat-Charvillon, N.: Algebraic side-channel attacks on the AES: why time also matters in DPA. In: Clavier, C., Gaj, K. (eds.) CHES, Lecture Notes in Computer Science, vol. 5747, pp. 97–111. Springer, Berlin (2009)
21. Bose, P.: private communication regarding thermal imaging
22. Shamir, A.,Tromer, E.: Acoustic cryptanalysis: On nosy people and noisy machines. http://people.csail.mit.edu/tromer/acoustic/
23. Ferrigno, J., Hlavac, M.: When AES blinks: introducing optical side channel, IET Information Security, vol. 2, 3rd edn. pp. 94–98 (2008)
24. Skorobogatov, S.P.: Using optical emission analysis for estimating contribution to power analysis. In: Breveglieri, L., Gueron, S., Koren, I., Naccache, D., Seifert, J.-P. (eds.) FDTC, IEEE Computer Society, pp. 111–119 (2009)
25. Boer, B. den, Lemke, K., Wicke, G.: A DPA attack against the modular reduction within a CRT implementation of RSA. In: Kaliski, B.S. Jr., Koç, Ç.K., Paar, C. (eds.): Cryptographic Hardware and Embedded Systems—CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13–15, 2002, Revised Papers, Lecture Notes in Computer Science, vol. 2523. pp. 228–243. Springer, Berlin (2003)
26. Jaffe, J.: Introduction to differential power analysis. In: Summer School on Cryptographic Hardware, Side-Channel and Fault Attacks, ECRYPT, pp. 42–45 (2006)

27. Jaffe, J.: A first-order DPA attack against AES in counter mode with unknown initial counter. In: Paillier, P., Verbauwhede, I. (eds.) CHES, Lecture Notes in Computer Science, vol. 4727, pp. 1–13. Springer, Berlin (2007)
28. Jaffe, J.: Using chosen messages to reduce DPA attack complexity (e.g. MISTY1) and to Amplify Leakage. CHES 2009 rump session presentation (2009)
29. Fouque, P.-A., Valette, F.: The doubling attack—why upwards is better than downwards. In: Walter, C.D., Koç, Ç.K., Paar, C. (eds.) CHES, Lecture Notes in Computer Science, vol. 2779, pp. 269–280. Springer, Berlin (2003)
30. Schindler, W.: A timing attack against RSA with the Chinese remainder theorem. In: Koç, Ç. K., Paar, C. (eds.): Cryptographic Hardware and Embedded Systems—CHES 2000, Second International Workshop, Worcester, MA, USA, August 17–18, 2000, Proceedings, Lecture Notes in Computer Science, vol. 1965, pp. 109–124. Springer, Berlin (2000)
31. Köpf, B., Basin, D.A.: An information-theoretic model for adaptive side-channel attacks. In: Ning, P., De Capitani Vimercati di, S., Syverson, P.F. (eds.) ACM Conference on Computer and Communications Security. ACM, pp. 286–296 (2007)
32. Veyrat-Charvillon, N., Standaert, F.-X.: Adaptive chosen-message side-channel attacks. In: Zhou, J., Yung, M. (eds.) ACNS, Lecture Notes in Computer Science, vol. 6123, pp. 186–199 (2010)
33. Clavier, C., Coron, J.-S., Dabbous, N.: Differential power analysis in the presence of hardware countermeasures. In: Koç, Ç.K., Paar, C. (eds.): Cryptographic Hardware and Embedded Systems—CHES 2000, Second International Workshop, Worcester, MA, USA, August 17–18, 2000, Proceedings, Lecture Notes in Computer Science, vol. 1965, pp.252–263. Springer, Berlin (2000)
34. Waddle, J., Wagner, D.: Towards efficient second-order power analysis. In: Joye, M., Quisquater, J.-J. (eds.): Cryptographic Hardware and Embedded Systems—CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11–13, 2004. Proceedings, Lecture Notes in Computer Science, vol. 3156, pp. 1–15. Springer, Berlin (2004)
35. Cryptography Research, Inc. DPA of SHA-1-based Key Derivation, March 2010. DPA Workstation Training
36. Jaffe, J.: DPA—what's now possible. CHES 2010 rump session presentation (2010)
37. Brier, E., Clavier, C., Olivier, F.: Correlation power analysis with a leakage model. In: Joye, M., Quisquater, J.-J. (eds.): Cryptographic Hardware and Embedded Systems—CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11–13, 2004. Proceedings, Lecture Notes in Computer Science, vol. 3156, pp. 16–29. Springer, Berlin (2004)
38. Gierlichs, B., Batina, L., Tuyls, P., Preneel, B. : Mutual information analysis. In: Oswald, E., Rohatgi, P. (eds.) CHES, Lecture Notes in Computer Science, vol. 5154, pp. 426–442. Springer, Berlin (2008)
39. Quisquater, J.-J., Samyde, D. : Electromagnetic analysis (EMA): measures and counter-measures for smart cards. In: Attali, I., Jensen, T.P. (eds.) E-smart, Lecture Notes in Computer Science, vol. 2140, pp. 200–210. Springer, Berlin (2001)
40. Gandolfi, K., Mourtel, C., Olivier, F. : Electromagnetic analysis: concrete results. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) CHES, Lecture Notes in Computer Science, vol. 2162, pp. 251–261. Springer, Berlin (2001)
41. Agrawal, D., Archambeault, B., Rao, J.R., Rohatgi, P.: The EM side-channel(s). In: Kaliski, B.S. Jr., Koç, Ç.K., Paar, C. (eds.): Cryptographic Hardware and Embedded Systems—CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13–15, 2002, Revised Papers, Lecture Notes in Computer Science, vol. 2523, pp. 29–45. Springer, Berlin (2003)
42. Bernstein, D.J.: Cache-timing attacks on AES. Technical report (2005)
43. Osvik, D.A., Shamir, A., Tromer, E.: Cache attacks and countermeasures: the case of AES. In: Topics in Cryptology—CT-RSA 2006, The Cryptographers Track at the RSA Conference 2006. pp. 1–20. Springer, Berlin (2005)
44. Aciiçmez, O., Koç, Ç.K., Seifert, J.-P.: Predicting secret keys via branch prediction. In: in Cryptology—CT-RSA 2007, The Cryptographers' Track at the RSA Conference 2007. p 225–242. Springer, Berlin (2007)
45. Chari S., Jutla, C.S., Rao, J.R., Rohatgi, P.: Towards sound approaches to counteract power-analysis attacks. In: Wiener, M.J. (ed): Advances in Cryptology—CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15–19, 1999, Proceedings, Lecture Notes in Computer Science, vol. 1666, pp. 398–412. Springer, Berlin (1999)
46. Messerges, T.S.: Using second-order power analysis to attack DPA resistant software. In: Koç, Ç.K., Paar, C. (eds.): Cryptographic Hardware and Embedded Systems—CHES 2000, Second International Workshop, Worcester, MA, USA, August 17–18, 2000, Proceedings, Lecture Notes in Computer Science, vol. 1965, pp. 238–251. Springer, Berlin (2000)
47. Joye, M., Paillier, P., Berry, S.: On second-order differential power analysis. In: Rao, J.R., Sunar, B. (eds.): Cryptographic Hardware and Embedded Systems—CHES 2005, 7th International Workshop, Edinburgh, UK, August 29–September 1, 2005, Proceedings, Lecture Notes in Computer Science, vol. 3659, pp. 293–308. Springer, Berlin (2005)
48. Oswald, E., Mangard, S., Herbst, C., Tillich, S.: Practical second-order DPA attacks for masked smart card implementations of block ciphers. In: Pointcheval, D. (ed.) CT-RSA, Lecture Notes in Computer Science, vol. 3860, pp. 192–207. Springer, Berlin (2006)
49. Prouff, E., Rivain, M., Bevan, R.: Statistical analysis of second order differential power analysis. IEEE Trans. Comput. **58**(6), 799–811 (2009)
50. Standaert, F.-X., Veyrat-Charvillon, N., Oswald, E., Gierlichs, B., Medwed, M., Kasper, M., Mangard, S.: The world is not enough: another look on second-order DPA. In: Abe, M. (ed.) ASIACRYPT, Lecture Notes in Computer Science, vol. 6477, pp. 112–129. Springer, Berlin (2010)
51. Chari, S., Rao, J.R., Rohatgi, P.: Template attacks. In: Kaliski, B.S. Jr., Koç, Ç.K., Paar, C. (eds.): Cryptographic Hardware and Embedded Systems—CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13–15, 2002, Revised Papers, Lecture Notes in Computer Science, vol. 2523, pp. 13–28. Springer, Berlin (2003)
52. Rechberger, C., Oswald, E.: Practical template attacks. In: Lim, C.H., Yung, M. (eds.) WISA, Lecture Notes in Computer Science, vol. 3325, pp. 440–456. Springer, Berlin (2004)
53. Schindler, W., Lemke, K., Paar, C.: A stochastic model for differential side channel cryptanalysis. In: Rao, J.R., Sunar, B. (eds.): Cryptographic Hardware and Embedded Systems—CHES 2005, 7th International Workshop, Edinburgh, UK, August 29–September 1, 2005, Proceedings, Lecture Notes in Computer Science, vol. 3659, pp. 30–46. Springer, Berlin (2005)
54. Novak, R.: Side-Channel Attack on Substitution Blocks. In: Zhou, J., Yung, M., Han, Y. (eds.) ACNS, Lecture Notes in Computer Science, vol. 2846, pp. 307–318. Springer, Berlin (2003)
55. Novak, R.: Sign-Based Differential Power Analysis. In: Chae, K., Yung, M. (eds.) WISA, Lecture Notes in Computer Science, vol. 2908, pp. 203–216. Springer, Berlin (2003)
56. Jaffe, J., Kocher, P., Jun, B.: Balanced cryptographic computational method and apparatus for leak minimizational in smartcards and other cryptosystems. US Patent 6,510,518
57. Jaffe, J., Kocher, P., Jun, B.: Hardware-level mitigation and DPA countermeasures for cryptographic devices. US Patent 6,654,884

58. Bystrov, A., Sokolov, D., Yakovlev, A., Koelmans, A.: Balancing power signature in secure systems. http://async.org.uk/ukasyncforum14/forum14-papers/forum14-bystrov.pdf (2003)
59. Tiri, K., Verbauwhede, I.: A logic level design methodology for a secure DPA resistant ASIC or FPGA implementation. In: DATE. IEEE Computer Society, pp. 246–251 (2004)
60. Sokolov, D., Murphy, J.P., Bystrov, A.V., Yakovlev, A.: Improving the security of dual-rail circuits. In: Joye, M., Quisquater, J.-J. (eds.): Cryptographic Hardware and Embedded Systems—CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11–13, 2004. Proceedings, Lecture Notes in Computer Science, vol. 3156, pp. 282–297. Springer, Berlin (2004)
61. Tiri, K., Verbauwhede, I.: Design method for constant power consumption of differential logic circuits. In: DATE. IEEE Computer Society, pp. 628–633 (2005)
62. Sokolov, D., Murphy, J.P., Bystrov, A.V., Yakovlev, A.: Design and analysis of dual-rail circuits for security applications. IEEE Trans. Comput. **54**(4), 449–460 (2005)
63. Aigner, M.J., Mangard, S., Menicocci, R., Olivieri, M., Scotti, G., Trifiletti, A.: A novel CMOS logic style with data-independent power consumption. In: International Symposium on Circuits and Systems (ISCAS 2005), 23–26 May 2005, pp. 1066–1069. IEEE, Kobe (2005)
64. Mace, F., Standaert, F.-X., Hassoune, I., Quisquater, J.-J., Legat, J.-D.: A dynamic current mode logic to counteract power analysis attacks. In: DCIS 2004—19th Conference on Design of Circuits and Integrated Systems. pp. 186–191. 11 (2004)
65. Deniz, Z.T., Leblebici, Y.: Low-power current mode logic for improved DPA-resistance in embedded systems. In: International Symposium on Circuits and Systems (ISCAS 2005), 23–26 May 2005, pp. 1059–1062. IEEE, Kobe (2005)
66. Moore, S.W., Mullins, R.D., Cunningham, P.A., Anderson, R.J., Taylor G.S.: Improving smart card security using self-timed circuits. In: ASYNC. p. 211. IEEE Computer Society (2002)
67. Yu, Z.C., Furber, S.B., Plana, L.A.: An investigation into the security of self-timed circuits. In: ASYNC. pp. 206–215. IEEE Computer Society (2003)
68. Rakers, P., Connell, L., Collins, T., Russell, D.: Secure contactless smartcard ASIC with DPA protection. IEEE J. Solid-State Circuits **36**(3), 559–565 (2001)
69. Ratanpal, G.B., Williams, R.D., Blalock, T.N.: An on-chip signal suppression counter measure to power analysis attacks. IEEE Trans. Dependable Sec. Comput. **1**(3), 179–189 (2004)
70. Kocher, P., Jaffe, J., Jun, B.: Using unpredictable information to minimize leakage from smartcards and other cryptosystems. US Patent 6,327,661
71. Kocher, P., Jaffe, J.: Secure modular exponentiation with leak minimization for smartcards and other cryptosystems. US Patent 6,298,442
72. Coron, J.-S.: Resistance against differential power analysis for elliptic curve cryptosystems. In: Koç, Ç.K., Paar, C. (eds.): Cryptographic Hardware and Embedded Systems, First International Workshop, CHES'99, Worcester, MA, USA, August 12–13, 1999, Proceedings, Lecture Notes in Computer Science, vol. 1717, pp. 292–302. Springer, Berlin (1999)
73. Kocher, P., Jaffe, J., Jun, B.: DES and other cryptographic, processes with leak minimization for smartcards and other cryptosystems. US Patent 6,278,783
74. Goubin, L., Patarin, J.: DES and differential power analysis (The "Duplication" Method). In: Koç Ç.K., Paar, C. (eds): Cryptographic Hardware and Embedded Systems, First International Workshop, CHES'99, Worcester, MA, USA, August 12–13, 1999, Proceedings, Lecture Notes in Computer Science, vol. 1717, pp. 158–172. Springer, Berlin (1999)
75. Prouff, E., Giraud, C., Aumônier, S.: Provably secure S-Box implementation based on Fourier transform. In: Goubin, L., Matsui, M. (eds.) CHES, Lecture Notes in Computer Science, vol. 4249, pp. 216–230. Springer, Berlin (2006)
76. Oswald, E., Mangard, S., Pramstaller, N., Rijmen, V.: A side-channel analysis resistant description of the AES S-Box. In: Gilbert, H., Handschuh, H. (eds.) FSE, Lecture Notes in Computer Science, vol. 3557, pp. 413–423. Springer, Berlin (2005)
77. Canright, D., Batina, L.: A very compact "Perfectly Masked" S-Box for AES. In: Bellovin, S.M., Gennaro, R., Keromytis, A.D., Yung, M. (eds) ACNS, Lecture Notes in Computer Science, vol. 5037, pp. 446–459 (2008)
78. Kocher, P.: Leak-resistant cryptographic indexed key update. US Patent 6,539,092
79. Kocher, P.: Design and validation strategies for obtaining assurance in countermeasures to power analysis and related attacks. NIST, physical security testing workshop edition, December 2005. http://csrc.nist.gov/groups/STM/cmvp/documents/fips140-3/physec/physecdoc.html
80. EMV2000: Integrated Circuit Card Specification for Payment Systems, Book 2—Security and Key Management, Appendix A1.3, December 2000. http://www.scardsoft.com/documents/EMV/EMV_2.pdf
81. Kocher, P., Rohatgi, P., Jaffe, J.: Verifiable leak resistant encryption and decryption, manuscript edition (2010) (To be posted at IACR ePrint archives)
82. Dziembowski, S., Pietrzak, K.: Leakage-resilient cryptography. In: FOCS. IEEE Computer Society, pp. 293–302 (2008)
83. Pietrzak, K.: A leakage-resilient mode of operation. In: Joux, A. (ed.) EUROCRYPT, Lecture Notes in Computer Science, vol. 5479 , pp. 462–482. Springer, Berlin (2009)
84. Katz, J., Vaikuntanathan, V.: Signature Schemes with Bounded Leakage Resilience. In: Matsui, M. (ed.) ASIACRYPT, Lecture Notes in Computer Science, vol. 5912, pp. 703–720. Springer, Berlin (2009)
85. Faust, S., Rabin, T., Reyzin, L., Tromer, E., Vaikuntanathan, V.: Protecting circuits from leakage: the computationally-bounded and noisy cases. In: Gilbert, H. (ed.) EUROCRYPT, Lecture Notes in Computer Science, vol. 6110, pp. 135–156. Springer, Berlin (2010)
86. Yu Y., Standaert, F.-X., Pereira, O., Yung M.: Practical leakage-resilient pseudorandom generators. In: Al-Shaer E., Keromytis, A.D., Shmatikov V. (eds.) ACM Conference on Computer and Communications Security. ACM, pp. 141–151 (2010)
87. Faust, S., Kiltz, E., Pietrzak, K., Rothblum, G.N.: Leakage-resilient signatures. In: Micciancio, D. (ed.) TCC, Lecture Notes in Computer Science, vol. 5978, pp. 343–360. Springer, Berlin (2010)
88. Dodis, Y., Pietrzak, K. : Leakage-resilient pseudorandom functions and side-channel attacks on feistel networks. In: Rabin, T. (ed.) CRYPTO, Lecture Notes in Computer Science, vol. 6223, pp. 21–40. Springer, Berlin (2010)
89. Center, L.: Workshop on provable security against physical attacks. http://www.lorentzcenter.nl/lc/web/2010/383/info.php3?wsid=383 (2010) Accessed Feb 2010
90. Standaert, F.-X., Pereira O., Yu Y., Quisquater, J.-J., Yung, M., Oswald E.: Leakage resilient cryptography in practice. Cryptology ePrint Archive, Report 2009/341. http://eprint.iacr.org/2009/341.ps (2009)