Towards Efficient Second-Order Power Analysis

Jason Waddle and David Wagner

University of California at Berkeley

Abstract. Viable cryptosystem designs must address power analysis attacks, and masking is a commonly proposed technique for defending against these side-channel attacks. It is possible to overcome simple masking by using higher-order techniques, but apparently only at some cost in terms of generality, number of required samples from the device being attacked, and computational complexity. We make progress towards ascertaining the significance of these costs by exploring a couple of attacks that attempt to efficiently employ second-order techniques to overcome masking. In particular, we consider two variants of second-order differential power analysis: Zero-Offset 2DPA and FFT 2DPA.

1 Introduction

Power analysis is a major concern for designers of smartcards and other embedded cryptosystems. The advance of Differential Power Analysis (DPA) in 1998 by Paul Kocher [1] made power analysis attacks even more practical since an attacker using DPA did not need to know very much about the device being attacked.

The technique of masking or duplication is commonly suggested as a way to stymie first-order power attacks, including DPA. In order to defeat masking, attacks would have to correlate the power consumption at multiple times during a single computation. Attacks of this sort were suggested and investigated (for example, by Thomas Messerges [2]), but it seems that the attacker was once again required to know significant details about the device under analysis.

This paper attempts to make progress towards a second-order analog of Differential Power Analysis. To this end, we suggest two second-order attacks, neither of which require much more time than straight DPA, but which are able to defeat some countermeasures. These attacks are basically preprocessing routines that attempt to correlate power traces with themselves and then apply standard DPA to the results.

In Section 2, we give some background and contrast first-order and second-order power analysis techniques. We also discuss the apparently inherent costs of higher-order attacks.

In Section 3, we present our model and give the intuition behind our techniques.

In Section 4, we give some techniques for second-order power analysis. In particular, we present some algorithms and analyze them in terms of limitations and requirements: generality, runtime, and number of required traces.

Section 5 contains some closing remarks, and Appendix A gives the formal derivations for the noise amplifications that are behind the limitations of the attacks in Section 4.

2 First-Order and Second-Order Power Analysis

We consider a cryptosystem that takes an input, performs some computations that combine this input and some internally stored secret, and produces an output. For concreteness, we will refer to this computation as an *encryption*, an input as a *plaintext*, the secret as a *key*, and the output as a *ciphertext*, though it is not necessary that the device actually be encrypting. An attacker would like to extract the secret from this device. If the attacker uses only the input and output information (i.e., the attacker treats the cryptosystem as a "black box"), it is operating in a traditional private-computation model; in this case, the secret's safety is entirely up to the algorithm implemented by the device.

In practice, however, the attacker may have access to some more *side-channel* information about the device's computation; if this extra information is correlated with the secret, it may be exploitable. This information can come from a variety of observables: timing, electromagnetic radiation, power consumption, etc. Since power consumption can usually be measured by externally probing the connection of the device with its power supply, it is one of the easiest of these side-channels to exploit, and it is our focus in this discussion.

2.1 First-Order Power Analysis Attacks

First-order attacks are characterized by the property that they exploit highly local correlation of the secret with the power trace. Typically, the secret-correlated power draw occurs at a consistent time during the encryption and has consistent sign and magnitude.

Simple Power Analysis (SPA) In simple first-order power analysis attacks, the adversary is assumed to have some fairly explicit knowledge of the analyzed cryptosystem. In particular, he knows the time at which the power consumption is correlated with part of the secret. By measuring the power consumption at this time (and perhaps averaging over a few encryptions to reduce the ambiguity introduced by noise), he gains some information about the key.

As a simple example, suppose the attacker knows that the first bit of the key k_0 is loaded into a register at 100μs into the encryption. The average power draw at 100μs is m, but when the k_0 is 0 this average is $m - \delta$ and when k_0 is 1, this average is $m + \delta$. Given enough samples of the power draw at 100μs to distinguish these means (where the number of samples required depends on the level of noise relative to δ), he can determine the first bit of the key.

Differential Power Analysis (DPA) One of the most amazing and troublesome features of differential power analysis is that, unlike with SPA, the attacker does not need such specific information about how the analyzed device implements its function. In particular, she can be ignorant of the specific times at which the power consumption is correlated with the secret; it is only necessary that the correlation is reasonably consistent.

In differential power analysis attacks, the attacker has identified some intermediate value in the computation that is 1) correlated with the power consumption, and 2) dependent only on the plaintext (or ciphertext or both) and some small part of the key. She gathers a collection of *power traces* by sampling power consumption at a very high frequency throughout a series of encryptions of different plaintexts. If the intermediate value is sufficiently correlated with the power consumption, the adversary can use the power traces to verify guesses at the small part of the key.

In particular, for each possible value of relevant part of the key, the attacker will divide the traces into groups according to the intermediate value predicted by current guess at the key and the trace's corresponding plaintext (or ciphertext); if the averaged power trace of each group differs noticeably from the others (the averaged differences will have a large difference at the time of correlation), it is likely that the current key guess is correct. Since incorrectly predicted intermediate value will not be correlated with the measured power traces, incorrect key guesses should result in all groups having very similar averaged power traces.

2.2 Higher-Order Attacks

A higher-order attack addresses a situation where there is some intermediate value (or set of values) that depends only on the plaintext and some small part of the key, but it is not correlated directly with the power consumption at any particular time. Instead, this value contributes to the joint distribution of the power consumption at a few times during the computation.

An important example of such a situation comes about when the *masking* (or duplication) technique is employed to protect against first-order attacks. As a typical example of masking, consider an implementation that wishes to perform a computation using some intermediate, key-dependent bit b. Rather than computing directly with b and opening itself up to DPA attacks, however, it performs the computation twice: once with a random bit r, then with the *masked bit* $(r + b)$.[1] The implementation is designed to use these two masked intermediate results as inputs to the rest of the computation.

In this case, knowledge of either r or $r + b$ alone is not of any use to the attacker. Since the first-order attacks look for local, linear correlation of b with the power draw, they are stymied. If, however, an attack could correlate the power

[1] Though we use the symbol '+' to denote the masking operation, we require nothing from it other than that $c = a + b$ implies $(-1)^c = (-1)^{a+b}$; for our purposes, it is convenient to just assume that '+' is exclusive-or.

consumption at the time r is present and the time $r+b$ is present (e.g., by multiplying the power consumptions at these times), it could gain some information on b.

For example, suppose a cryptographic device employs masking to hide some intermediate bit b that is derived directly from the key, but displays the following behavior: at 100μs, the average power draw is $m + \delta(-1)^r$ and at 210μs it is $m + \delta(-1)^{(r+b)}$. An attacker aware of this fact could multiply the samples at these times for each trace and obtain a product value with expected value[2]

$$\mathbb{E}[\text{product of samples}] = \begin{cases} m^2 + \delta^2 & \text{if } b = 0 \\ m^2 - \delta^2 & \text{if } b = 1 \end{cases} \tag{1}$$

Summing the samples over n encryptions, the means would be $n(m^2+\delta^2)$ for $b = 0$ and $n(m^2-\delta^2)$ for $b = 1$. By choosing n large enough to reduce the relative effect of noise, the attacker could distinguish these distributions and deduce b. An attack of this sort is the second-order analog of an SPA attack.

But how practical is this really? A higher-order attack seems to face two major problems:

- How much does the process of correlation amplify the noise, thereby increasing standard deviation and requiring more samples to reliably differentiate distributions?
- How does it identify the times when the power consumption is correlated with an intermediate value?

The first issue is apparent when calculating the standard deviation of the product computed in the above attack. If the power consumption at times 100μs and 210μs both have standard deviation σ, then the product has standard deviation

$$\sigma_{\text{product}} = \begin{cases} \sqrt{\sigma^4 + 2\sigma^2 m^2 + 4\delta^2 m^m + 2\delta^2\sigma^2} & \text{if } b = 0, \\ \sqrt{\sigma^4 + 2\sigma^2 m^2 + 2\delta^2\sigma^2} & \text{if } b = 1 \end{cases} \tag{2}$$

effectively squaring the standard deviation of zero-mean noise. This means that substantially many more samples are required to distinguish the $b = 0$ and $b = 1$ distributions than would be required in a first-order attack, if one were possible.

The second issue is essentially the higher-order analog of the problem with SPA: attackers require exact knowledge of the time at which the intermediate value and the power consumption are correlated. DPA resolves this problem by considering many samples of the power consumption throughout an encryption. Unfortunately, the natural generalization of this approach to even second-order attacks, where a product would be accumulated for each (t_1, t_2) time pair, is extremely computationally taxing. The second-order attacks discussed in this paper avoid this overhead.

[2] Here the expectation is taken over both the random noise and the value of the masking bit r, and the noise components at times 100μs and 210μs are assumed independent.

3 The Power Analysis Model

Both of the attacks we present are second-order attacks which are essentially preprocessing steps applied to the power traces followed by standard DPA.

In this section, we develop our model and present standard DPA in this framework, both as a point of reference and as a necessary subroutine for our attacks, which are described in Section 4.

3.1 The Model

We assume that the attacker has guessed part of the key and has predicted an intermediate bit value b for each of the power traces, grouping them into a $b = 0$ and a $b = 1$ group. For simplicity, we assume there are n traces in each of these groups: trace i from group b is called T_i^b, where $0 \leq i < n$. Each trace contains samples at m evenly spaced times; the sample at time t from this trace is denoted $T_i^b(t)$, where $0 \leq t < m$.

Each sample has a *noise* component and possibly a *signal* component, if it is correlated with b. We assume that each noise component is Gaussian with equal standard deviation and independent of the noise in other samples in its own trace and other traces. For simplicity, we also assume that the input has been normalized so that each noise component is a 0-mean Gaussian with standard deviation one (i.e., $\sim \mathcal{N}(0, 1)$). The random variable for the noise component in trace i from group b at time t is $S_i^b(t)$, for $0 \leq t < m$.

We assume that the device being analyzed is utilizing masking so that there is a uniformly distributed independent random variable for each trace that corresponds to the masking bit; it will be more convenient for us to deal with $\{\pm 1\}$ bit values, so if the random bit in trace i from group b is r, we define the random variable $R_i^b = (-1)^r$.

Finally, if the guess for b is correct, the power consumption is correlated with the random masking bit and the intermediate value b at the same times in each trace. Specifically, we assume that there is some parameter d (in units of the standard deviation of the noise) and times c_0 and c_1 such that the random bit makes a contribution of dR_i^b to the power consumption at time c_0 and the masked bit makes a contribution of $d(-1)^b R_i^b$ at time c_1.

We can now characterize the trace sample distributions in terms of these noise and signal components:

– If the guess of the key is correct, then for $0 \leq i < n$, $0 \leq t < m$, and $b \in \{0, 1\}$, we have:

$$T_i^b(t) = \begin{cases} S_i^b(t) + dR_i^b & \text{if } t = c_0 \\ S_i^b(t) + d(-1)^b R_i^b & \text{if } t = c_1 \\ S_i^b(t) & \text{otherwise} \end{cases} \tag{3}$$

– If the key is predicted incorrectly, however, then the groups are not correlated with the true value of b in each trace and hence there is no correlation between the grouping and the power consumption in the traces, so, for $0 \leq i < n$, $0 \leq t < m$, and $b \in \{0, 1\}$:

$$T_i^b(t) = S_t^b(t) \tag{4}$$

Given these traces as inputs, the algorithms try to decide whether the groupings (and hence the guess for the key) are correct by distinguishing these distributions.

3.2 The Generic DPA Subroutine

Both algorithms use a subroutine `DPA` after their preprocessing step. For our purposes, this subroutine simply takes the two groups of traces, T^0 and T^1, a threshold value τ, and determines whether the groups' totalled traces differ by more than τ at any sample time. If the difference of the totalled traces is greater than τ at any point, `DPA` returns 1, indicating that T^0 and T^1 have different distributions; if the difference is no more than τ at any point, `DPA` returns 0, indicating that it thinks T^0 and T^1 are identically distributed.

`DPA`(T^0, T^1, τ)
    for each $t \in \{0, \ldots, m-1\}$:
        $s \leftarrow 0$
        for each $i \in \{0, \ldots, n-1\}$:
            $s \leftarrow s + T_i^0(t) - T_i^1(t)$
        if $|s| > \tau$ return 1
    return 0

When using the `DPA` subroutine, it is most important to pick the threshold, τ, appropriately. Typically, to minimize the impact of false positives and false negatives, τ should be half the difference. This is perhaps unexpected since false positives are actually far more likely than false negatives when using a midpoint threshold test since false positives can occur if *any* of the m times' samples sum deviates above τ, while false negatives require exactly the correlated time's samples to deviate below τ. The reason for not choosing τ to equalize the probabilities is that false negatives are far more detrimental than false positives: an attack suggesting two likely subkeys is more helpful than an attack suggesting none.

An equally important consideration in using `DPA` is whether τ is large enough compared to the noise to reduce the probability of error. Typically, the samples' noise components will be independent and the summed samples' noise will be Gaussian, so we can can achieve negligible probability of error by using n large enough that τ is some constant multiple of the standard deviation.

`DPA` runs in time $\Theta(nm)$. Each run of `DPA` decides the correctness of only one guessed grouping, however, so an attack that tries l groupings runs in time $\Theta(nml)$.

4 Our Second-Order Attacks

The two second-order variants of DPA that we discuss are Zero-Offset 2DPA and FFT 2DPA. The former is applied in the special but not necessarily unlikely situation when the power correlation times for the two bits are coincident (i.e., the random bit r and the masked bit $r+b$ are correlated with the power consumption at the same time). The latter attack applies to the more general situation where the attacker does not know the times of correlation; it discovers the correlation with only slight computational overhead but pays a price in the number of required samples.

4.1 Zero-Offset 2DPA

Zero-Offset 2DPA is a very simple variation of ordinary first-order DPA that can be applied against systems that employ masking in such a way that both the random bit r and the masked intermediate bit $r+b$ correlate with the power consumption at the same time. In the language of our model, $c_0 = c_1$.

The coincident effect of the two masked values may seem to be too specialized of a circumstance to occur in practice, but it does come up. The motivation for this attack is the claim by Coron and Goubin [3] that some techniques suggested by Messerges [4] were insecure due to some register containing the multi-bit intermediate value a or its complement $\overline{a}$. Since Messerges assumes a power consumption model based on Hamming weight, it was not clear how a first-order attack would exploit this register. However, we observe that such a system can be attacked (even in the Hamming model) by a Zero-Offset 2DPA that uses as its intermediate value the exclusive-or of the first two bits of a. Another example of a situation with coincident power consumption correlation is in a paired circuit design that computes with both the random and masked inputs in parallel.

Combining $c_0 = c_1$ with Equation (3), we see that in a correct grouping:

$$T_i^b(t) = \begin{cases} S_i^b(t) + dR_i^b + d(-1)^b R_i^b & \text{if } t = c_0 \\ S_i^b(t) & \text{otherwise} \end{cases} \tag{5}$$

In an incorrect grouping, $T_i^b(t)$ is distributed exactly as in the general uncorrelated case in Equation (4).

Note that in a correct grouping, when $b = 1$, the influence of the two bits cancel, leaving $T_i^1(c_0) = S_i^1(c_0)$, while when $b = 0$, the influences of the two bits combine constructively and we get $T_i^0(c_0) = S_i^0(c_0) + 2dR_i^0$. In the former case, there appears to be no influence of the bits on the power consumption distribution, but in the latter case, the bits contribute a bimodal component. The bimodal component has mean 0, however, so it would not be apparent in a first-order averaging analysis.

Zero-offset 2DPA exploits the bimodal component for the $b = 0$ case by simply squaring the samples in the power traces before running straight DPA.

`Zero-Offset-2DPA`(T^0, T^1)

for each $b \in \{0,1\}$, $i \in \{0,\ldots,n\}$, $t \in \{0,\ldots,m\}$:

  $T_i^b(t) \leftarrow (T_i^b(t))^2$

return `DPA`$(T^0, T^1, 2nd^2)$

Why does this work? Suppose we have a correct grouping and consider the expected values for the sum of the squares of the samples at time c_0 in the two groups:

- if $b = 0$,

$$\begin{aligned}\mathbb{E}\left[\sum_{i=0}^{n-1}[T_i^0(c_0)]^2\right] &= \sum_{i=0}^{n-1}\mathbb{E}[(S_i^0(c_0))^2 + 4dS_i^0(c_0)R_i^0 + 4d^2(R_i^0)^2] \\ &= \sum_{i=0}^{n-1}\left(\mathbb{E}[(S_i^0(c_0))^2] + \mathbb{E}[4dS_i^0(c_0)R_i^0] + \mathbb{E}[4d^2(R_i^0)^2]\right) \\ &= \sum_{i=0}^{n-1}(1 + 0 + 4d^2) \\ &= 4nd^2 + n \end{aligned} \tag{6}$$

- if $b = 1$,

$$\begin{aligned}\mathbb{E}\left[\sum_{i=0}^{n-1}[T_i^1(c_0)]^2\right] &= \sum_{i=0}^{n-1}\mathbb{E}[(S_i^1(c_0))^2] \\ &= \sum_{i=0}^{n-1}1 \\ &= n \end{aligned} \tag{7}$$

The above derivations use the fact that if $S \sim \mathcal{N}(0,1)$ then $S^2 \sim \chi^2(1,0)$ (i.e., S^2 has χ^2 distribution with $\nu = 1$ degree of freedom and non-centrality parameter $\delta^2 = 0$), and the expected value of a $\chi^2(\nu, \delta^2)$ random variable is $\nu + \delta^2$.

Thus, the expected difference of the sum of products for the c_0 samples is $4nd^2$, while the expected difference for incorrect groupings is clearly 0. In Section A.1, we show that the difference of the groups' sums of products is essentially Gaussian with standard deviation

$$\sigma = \sqrt{n(16d^2 + 4)}. \tag{8}$$

For an attack that uses a DPA threshold value at least k standard deviations from the mean, we will need at least $k^2 \cdot \frac{(4d^2+1)}{4d^4}$ traces. This $\frac{(4d^2+1)}{4d^4}$ blowup factor may be substantial; recall that d is in units of the standard deviation of the noise, so it may be significantly less than 1.

The preprocessing in `Zero-Offset-DPA` takes time $\Theta(nm)$. After this preprocessing, each of l subsequent guessed groupings can be tested using `DPA` in time $\Theta(nm)$, for a total runtime of $\Theta(nm + nml) = \Theta(nml)$. It is important to keep in mind when comparing these run times that the number n of required traces for `Zero-Offset-DPA` can be somewhat larger than would be necessary for first-order DPA—if a first-order attack were possible.

A Natural Variation: Known-Offset 2DPA If the difference $s = c_1 - c_0$ is non-zero but known, a similar attack may be mounted. Instead of calculating the squares of the samples, the adversary can calculate the lagged product:

$$L_i^b(t, s) = T_i^b(t) \cdot T_i^b(t + s), \tag{9}$$

where the addition $t + s$ is intended to be cyclic in $\{0, \ldots n - 1\}$.

This lagged product at the correct offset $s = c_1 - c_0$ has properties similar to the squared samples discussed above, and can be used in the same way.

4.2 FFT 2DPA

Fast Fourier Transform (FFT) 2DPA is useful in that it is more general than Zero-Offset 2DPA: it does not require that the times of correlation be coincident, and it does not require any particular information about c_0 and c_1.

To achieve this, it uses the FFT to compute the *correlation* of a trace with itself—an *autocorrelation*. The autocorrelation A_i^b of a trace T_i^b is also defined on values $t \in \{0, \ldots, m - 1\}$, but this argument is considered an *offset* or *lag* value rather than an absolute time. Specifically, for $b \in \{0, 1\}$, $0 \leq i < n$, and $0 \leq t < m$,

$$A_i^b(t) = \sum_{j=0}^{m-1} T_i^b(j) \cdot T_i^b(j + t) \tag{10}$$

The argument $t + j$ in $T_i^b(j + t)$ is understood to be cyclic in $\{0, \ldots, m - 1\}$, so that $A_i^b(t) = A_i^b(m - t)$, and we really only need to consider $0 \leq t \leq m/2$.

To see why $A_i^b(t)$ might be useful, recall Equation (3) and notice that most of the terms of $A_i^b(t)$ are of the form $S_i^b(j) \cdot S_i^b(j + t)$; in fact, the only terms that differ are where j or $j + t$ is c_0 or c_1. This observation suggests a way to view the sum for $A_i^b(t)$ by splitting it up by the different types of terms from Equation (3), and in fact it is instructive to do so. To simplify notation, let $Q = \{c_0 - t, c_0, c_1 - t, c_1\}$, the set of "interesting" indices, where the terms of $A_i^b(t)$ are "unusual" when $j \in Q$. Assuming $t \neq c_1 - c_0$,

$$\begin{aligned} A_i^b(t) = {} & S_i^b(c_0 - t) \cdot [S_i^b(c_0) + dR_i^b] \\ & + [S_i^b(c_0) + dR_i^b] \cdot S_i^b(c_0 + t) \\ & + S_i^b(c_1 - t) \cdot [S_i^b(c_1) + d(-1)^b R_i^b] \\ & + [S_i^b(c_1) + d(-1)^b R_i^b] \cdot S_i^b(c_1 + t) \\ & + \sum_{j \notin Q} S_i^b(j) \cdot S_i^b(j + t) \end{aligned} \tag{11}$$

and we can distribute and recombine terms to get

$$\begin{aligned} A_i^b(t) &= [S_i^b(c_0 - t) + S_i^b(c_0 + t)] \cdot dR_i^b \\ &\quad + [S_i^b(c_1 - t) + S_i^b(c_1 + t)] \cdot d(-1)^b R_i^b \\ &\quad + \sum_{j=0}^{m-1} S_i^b(j) \cdot S_i^b(j + t). \end{aligned} \tag{12}$$

Using Equation (12) and the fact that $\mathbb{E}[XY] = \mathbb{E}[X] \cdot \mathbb{E}[Y]$ when X and Y are independent random variables, it is straightforward to verify that $\mathbb{E}[A_i^b(t)] = 0$ when $t \neq c_1 - c_0$; its terms in that case are products involving some 0-mean independent random variable (this is exactly what we show in Equation (15)). On the other hand, $A_i^b(c_1 - c_0)$ involves terms that are products of dependent random variables, as can be seen by reference to Equation (10). We make frequent use of Equation (12) in our derivations in this section and in Appendix A.2.

This technique requires a subroutine to compute the autocorrelation of a trace:

`Autocorrelate`(T)
1 : $F \leftarrow$ `FFT`(T)
2 : for each $t \in \{0, \ldots, m-1\}$:
3 : $\quad F(t) \leftarrow |F(t)|^2$
4 : return `Inv-FFT`(F)

The $|F(t)|^2$ in line 3 is the squared $\mathcal{L}_2$-norm of the complex number $F(t)$ (i.e., $|F(t)|^2 = \overline{F(t)} \cdot F(t)$, where $\overline{\alpha}$ denotes the complex conjugate of α).

The subroutine `FFT` computes the usual Discrete Fourier Transform:

$$(\texttt{FFT}(T))(x) = \sum_{j=0}^{m-1} T(j) \cdot \omega^{-xj} \tag{13}$$

and `Inv-FFT` computes the Inverse Discrete Fourier Transform:

$$(\texttt{Inv-FFT}(T))(y) = \frac{1}{m} \sum_{j=0}^{m-1} T(j) \cdot \omega^{xj} \tag{14}$$

In the above equations, ω is a complex primitive mth root of unity (i.e., $\omega \in \mathbb{C}$, $\omega^m = 1$, and $\omega^k \neq 1$ for all $0 < k < m$).

The subroutines `FFT`, `Inv-FFT`, and therefore `Autocorrelate` itself all run in time $\Theta(m \log m)$.

We can now define the top-level `FFT-2DPA` algorithm:

`FFT-2DPA`(T^0, T^1, τ)
for each $b \in \{0, 1\}$, $t \in \{0, \ldots, m-1\}$:
$\quad Z^b(t) \leftarrow 0$
for each $b \in \{0, 1\}$, $i \in \{0, \ldots, n-1\}$:
$\quad A^b \leftarrow$ `Autocorrelate`(T_i^b)

for each $t \in \{0, \ldots, m-1\}$:
    $Z^b(t) \leftarrow Z^b(t) + A^b(t)$
return $\mathtt{DPA}(Z^0,\, Z^1,\, nd^2)$

What makes this work? Assuming a correct grouping, the expected sums are:

– $t \neq c_1 - c_0$:

$$\begin{aligned}
\mathbb{E}[Z^b(t)] &= \mathbb{E}\left[\sum_{i=0}^{n-1}\sum_{j=0}^{m-1}(T_i^b(j)\cdot T_i^b(j+t))\right] \\
&= n\sum_{j=0}^{m-1}\mathbb{E}[T_0^b(j)\cdot T_0^b(j+t)] \\
&= n\,\mathbb{E}\left[[S_0^b(c_0-t)+S_0^b(c_0+t)]\cdot dR_0^b\right] \\
&+ n\,\mathbb{E}\left[[S_0^b(c_1-t)+S_0^b(c_1+t)]\cdot d(-1)^b R_0^b\right] \\
&+ nm\,\mathbb{E}\left[S_0^b(0)\cdot S_0^b(0+t)\right] \\
&= 0
\end{aligned} \tag{15}$$

– $t = (c_1 - c_0)$:

$$\begin{aligned}
\mathbb{E}[Z^b(t)] &= \mathbb{E}\left[\sum_{i=0}^{n-1}\sum_{j=0}^{m-1}(T_i^b(j)\cdot T_i^b(j+t))\right] \\
&= n\sum_{j=0}^{m-1}\mathbb{E}[T_0^b(j)\cdot T_0^b(j+t)] \\
&= n\,\mathbb{E}\left[[S_0^b(c_0-t)+S_0^b(c_1)]\cdot dR_0^b\right] \\
&+ n\,\mathbb{E}\left[[S_0^b(c_0)+S_0^b(c_1+t)]\cdot d(-1)^b R_0^b\right] \\
&+ n\,\mathbb{E}[d^2(R_0^b)^2(-1)^b] \\
&+ nm\,\mathbb{E}\left[S_0^b(0)\cdot S_0^b(0+t)\right] \\
&= 0+0+n\,\mathbb{E}[d^2(R_0^b)^2(-1)^b]+0 \\
&= nd^2(-1)^b
\end{aligned} \tag{16}$$

So in a correct grouping, we have

$$\mathbb{E}[Z^0(t)-Z^1(t)] = \begin{cases} 2nd^2 & \text{if } t = c_1 - c_0 \\ 0 & \text{otherwise.} \end{cases} \tag{17}$$

In incorrect groupings, however, $\mathbb{E}[Z^0(t)-Z^1(t)] = 0$ for all $t \in \{0, \ldots, m-1\}$.

In Section A.2, we see that this distribution is closely approximated by a Gaussian with standard deviation $\sigma = \sqrt{n(8d^2+2m)}$, so that an attacker who wishes to use a threshold at least k standard deviations away from the mean needs n to be at least about $k^2 \cdot \frac{(4d^2+m)}{2d^4}$.

Note that the noise from the other samples contributes significantly to the standard deviation at $Z^b(c_1 - c_0)$, so this attack would only be practical for relatively short traces and a significant correlated bit influence (i.e., when m is small and d is not much smaller than 1).

The preprocessing in `FFT-2DPA` runs in time $\Theta(nm \log m)$. After this preprocessing, however, each of l guessed groupings can be tested using `DPA` in time $\Theta(nm)$, for a total runtime of $\Theta(nm \log m + nml)$, amortizing to $\Theta(nml)$ if $l = \Omega(\log m)$. Again, when considering this runtime, it is important to keep in mind that the number n of required traces can be substantially larger than would be necessary for first-order DPA—if a first-order attack were possible.

FFT and Known-Offset 2DPA It might be very helpful in practice to use the FFT in second-order power analysis attacks for attempting to determine the offset of correlation. With a few traces, it could be possible to use an FFT to find the offset s of repeated computations, such as when the same function is computed with the random bit r at time c_0 and with the masked bit $r + b$ at time $c_0 + s$.

With even a few values of s suggested by an FFT on these traces, a Known-Offset 2DPA attack could be attempted, which could require far fewer traces than straight FFT 2DPA since Known-Offset 2DPA suffers from less noise amplification.

5 Conclusion

We explored two second-order attacks that attempt to defeat masking while minimizing computation resource requirements in terms of space and time.

The first, Zero-Offset 2DPA, works in the special situation where the masking bit and the masked bit are coincidentally correlated with the power consumption, either canceling out or contributing a bimodal component. It runs with almost no noticeable overhead over standard DPA, but the number of required power traces increases more quickly with the relative noise present in the power consumption.

The second technique, FFT 2DPA, works in the more general situation where the attacker knows very little about the device being analyzed and suffers only logarithmic overhead in terms of runtime. On the other hand, it also requires many more power traces as the relative noise increases.

In summary, we expect that Zero-Offset 2DPA and Known-Offset 2DPA can be of some practical use, but FFT 2DPA probably suffers from too much noise amplification to be generally effective. However, if the traces are fairly short and the correlated bit influence fairly large, it can be effective.

References

1. Paul Kocher, Joshua Jaffe, and Benjamin Jun, "Differential Power Analysis," in proceedings of *Advances in Cryptology—CRYPTO '99*, Springer-Verlag, 1999, pp. 388-397.
2. Thomas Messerges, "Using Second-Order Power Analysis to Attack DPA Resistant Software," *Lecture Notes in Computer Science*, 1965:238-??, 2001.
3. Jean-Sébastien Coron and Louis Goubin, "On Boolean and Arithmetic Masking against Differential Power Analysis", in *Proceedings of Workshop on Cryptographic Hardware and Embedded Systems*, Springer-Verlag, August 2000.
4. Thomas S. Messerges, "Securing the AES Finalists Against Power Analysis Attacks," in *Proceedings of Workshop on Cryptographic Hardware and Embedded Systems*, Springer-Verlag, August 1999, pp. 144-157.

A Noise Amplification

In this section, we attempt to characterize the distribution of the estimators that we use to distinguish the target distributions. In particular, we show that the estimators have near-Gaussian distributions and we calculate their standard deviations.

A.1 Zero-Offset 2DPA

As in Section 4.1, we assume that the times of correlation are coincident, so that $c_0 = c_1$. From this, we get that the distribution of the samples in a correct grouping follows Equation (5):

$$T_i^b(t) = \begin{cases} S_i^b(t) + dR_i^b + d(-1)^b R_i^b & \text{if } t = c_0 \\ S_i^b(t) & \text{otherwise} \end{cases} \tag{18}$$

The sum

$$\sum_{i=0}^{n-1} [T_i^0(c_0)]^2 = \sum_{i=0}^{n-1} [S_i^0(c_0) + 2dR_i^0]^2 \tag{19}$$

is then a $\chi^2(\nu, \delta^2)$-distributed random variable with $\nu = n$ degrees of freedom and non-centrality parameter $\delta^2 = \sum_{i=0}^{n-1}(2dR_i^0)^2 = 4nd^2$. It has mean $\nu + \delta^2 = 4nd^2 + n$ and standard deviation $\sqrt{2(\nu + 2\delta^2)} = \sqrt{2(n + 8nd^2)} = \sqrt{n(16d^2 + 2)}$.

A common rule of thumb is that χ^2-distributed random variables with over thirty degrees of freedom are closely approximated by Gaussians. We expect $n \gg 30$, so we say

$$\sum_{i=0}^{n-1} [T_i^0(c_0)]^2 \sim \mathcal{N}\left(4nd^2 + n, \sqrt{n(16d^2 + 2)}\right). \tag{20}$$

Similarly, we obtain $\sum_{i=0}^{n-1}[T_i^1(c_0)]^2 \sim \chi^2(n,0)$, which, since $n \gg 30$, we approximate with

$$\sum_{i=0}^{n-1}[T_i^1(c_0)]^2 \sim \mathcal{N}\left(n, \sqrt{2n}\right). \tag{21}$$

The difference of the summed squares is then

$$\sum_{i=0}^{n-1}\left[(T_i^0(c_0))^2 - (T_i^1(c_0))^2\right] \sim \mathcal{N}\left(4nd^2, \sqrt{n(16d^2+4)}\right). \tag{22}$$

A.2 FFT 2DPA

Recalling our discussion from Section 4.2, we want to examine the distribution of

$$Z^b(t) = \sum_{i=0}^{n-1}\sum_{j=0}^{m-1} T_i^b(j) \cdot T_i^b(j+t). \tag{23}$$

when $t = c_1 - c_0$. Its standard deviation should dominate that of $Z^b(t')$ for $t' \neq c_1 - c_0$ (for simplicity, we assume $c_1 - c_0 \neq c_0 - c_1$).

In Section 4.2, we saw that $\mathbb{E}[Z^b(t)] = nd^2(-1)^b$. We would now like to calculate its standard deviation.

In the following, we liberally use the fact that

$$\mathrm{Var}[X+Y] = \mathrm{Var}[X] + \mathrm{Var}[Y] + 2\,\mathrm{Cov}[X,Y], \tag{24}$$

where $\mathrm{Cov}[X,Y]$ is the *covariance* of X and Y ($\mathrm{Cov}[X,Y] \triangleq \mathbb{E}[XY] - \mathbb{E}[X]\,\mathbb{E}[Y]$). We would often like to add variances of random variables that are not independent; Equation (24) says we can do so if the random variables have 0 covariance.

Since the traces are independent and identically distributed,

$$\begin{aligned}
\mathrm{Var}[Z^b(t)] &= \sum_{i=0}^{n-1} \mathrm{Var}\left[\sum_{j=0}^{m-1} T_i^b(j)\cdot T_i^b(j+t)\right] \\
&= n\,\mathrm{Var}\left[\sum_{j=0}^{m-1} T_0^b(j)\cdot T_0^b(j+t)\right] \\
&= n\,\mathrm{Var}\left[dR_0^b([S_i^b(c_0-t) + S_0^b(c_1)] + (-1)^b[S_0^b(c_0) + S_0^b(c_1+t)])\right] \\
&\quad + n\,\mathrm{Var}\left[\sum_{j=0}^{m-1} S_0^b(j)\cdot S_0^b(j+t)\right]
\end{aligned} \tag{25}$$

where we were able to split the variance in the last line since the two terms have 0 covariance.

To calculate $\mathrm{Var}\left[dR_0^b([S_0^b(c_0-t)+S_0^b(c_1)]+(-1)^b[S_0^b(c_0)+S_0^b(c_1+t)])\right]$, note that its terms have 0 covariance. For example:

$$\begin{aligned}\mathrm{Cov}[dR_0^bS_0^b(c_0-t), dR_0^bS_0^b(c_1)] &= \mathbb{E}[(dR_0^b)^2S_0^b(c_0-t)\cdot S_0^b(c_1)] \\ &\quad - \mathbb{E}[dR_0^bS_0^b(c_0-t)]\,\mathbb{E}[dR_0^bS_0^b(c_1)] \\ &= 0-0=0 \end{aligned} \tag{26}$$

since the expectation of a product involving an independent 0-mean random variable is 0. Furthermore, it is easy to check that each term has the same variance, and

$$\begin{aligned}\mathrm{Var}[dR_0^bS_0^b(c_1)] &= \mathbb{E}\left[[dR_0^bS_0^b(c_1)]^2\right] - \mathbb{E}[dR_0^bS_0^b(c_1)]^2 \\ &= d^2\,\mathbb{E}\left[[S_0^b(c_1)]^2\right] - 0 \\ &= d^2, \end{aligned} \tag{27}$$

for a total contribution of

$$\mathrm{Var}\left[dR_0^b([S_0^b(c_0-t)+S_0^b(c_1)]+(-1)^b[S_0^b(c_0)+S_0^b(c_1+t)])\right] = 4d^2. \tag{28}$$

The calculation of $\mathrm{Var}\left[\sum_{j=0}^{m-1} S_0^b(j)\cdot S_0^b(j+t)\right]$ is similar since its terms also have covariance 0 and they all have the same variance. Thus,

$$\begin{aligned}\mathrm{Var}\left[\sum_{j=0}^{m-1} S_0^b(j)S_0^b(j+t)\right] &= m\,\mathrm{Var}[S_0^b(0)S_0^b(0+t)] \\ &= m\left(\mathbb{E}\left[[S_0^b(0)]^2[S_0^b(t)]^2\right] - \mathbb{E}[S_0^b(0)S_0^b(t)]^2\right) \\ &= m(1+0) = m. \end{aligned} \tag{29}$$

Finally, plugging Equations (28) and (29) into Equation (25), we get the result

$$\mathrm{Var}[Z^b(t)] = n(m+4d^2) \tag{30}$$

and the corresponding standard deviation is $\sqrt{n(m+4d^2)}$.

As in Section A.1, we expect n to be large and we say

$$Z^b(t) \sim \mathcal{N}\left(nd^2(-1)^b, \sqrt{n(4d^2+m)}\right). \tag{31}$$

Finally, we get the distribution of the difference:

$$Z^0(t) - Z^1(t) \sim \mathcal{N}\left(2nd^2, \sqrt{n(8d^2+2m)}\right). \tag{32}$$