Using Second-Order Power Analysis to Attack DPA Resistant Software

Thomas S. Messerges

Motorola Labs, Motorola
1301 E. Algonquin Road, Room 2712, Schaumburg, IL 60196
Tom.Messerges@motorola.com

Abstract. Under a simple power leakage model based on Hamming weight, a software implementation of a data-whitening routine is shown to be vulnerable to a first-order Differential Power Analysis (DPA) attack. This routine is modified to resist the first-order DPA attack, but is subsequently shown to be vulnerable to a second-order DPA attack. A second-order DPA attack that is optimal under certain assumptions is also proposed. Experimental results in an ST16 smartcard confirm the practicality of the first and second-order DPA attacks.

1 Introduction

Recently there has been increased concern over the vulnerabilities of cryptographic algorithms to leakage attacks [1]. These attacks exploit the fact that a hardware device can sometimes leak information when running a cryptographic algorithm. One source of leaked information is the time-varying power consumption of a device executing a cryptographic algorithm. In 1998, Kocher et al. introduced a leakage attack that uses a technique called Differential Power Analysis (DPA) [2]. Attacks using DPA have been shown to be quite successful at breaking the security of smartcards [3]. Researchers have reported power analysis attack against many algorithms, [e.g., 4-7] and have also developed countermeasures that can resist such attacks [e.g., 8-9].

The main focus of past research has been on first-order DPA attacks. However, higher-order DPA attacks [2] also need to be understood. For example, countermeasures that prevent first-order DPA attacks may not be effective against higher-order attacks. In my investigations, I assume that power leakage can be described by a simple model based on Hamming weights. I use this model to show that a naive implementation of a data-whitening routine is vulnerable to a first-order DPA attack. I then implement a countermeasure to protect this routine from attack, but this new routine is subsequently shown to be vulnerable to a second-order DPA attack. Finally, I show that this second-order DPA attack is approximately optimal under certain reasonable assumptions. Experimental results in an ST16 smartcard manufactured by ST Microelectronics are used to confirm the practicality of my attacks.

1.1 Definitions

A higher-order DPA attack is defined by Kocher et al. [2] as a DPA attack that combines one or more samples within a single power trace. During a first-order DPA attack, the

Ç.K. Koç and C.Paar (Eds.): CHES 2000, LNCS 1965, pp. 238 - 251, 2000.

attacker monitors power consumption signals and calculates the individual statistical properties of the signals at each sample time. In a higher-order DPA attack, the attacker calculates joint statistical properties of the power consumption at multiple sample times within the power signals. For the purpose of this paper, the definition of an *n*th-order DPA attack is given as follows.

Definition 1. An ***nth-order DPA attack*** *makes use of n different samples in the power consumption signal that correspond to n different intermediate values calculated during the execution of an algorithm.*

The attacks described in this paper are proven to be *sound*. The definition of a sound DPA attack is given as follows.

Definition 2. A DPA attack against an algorithm's secret key is ***sound*** *when it is theoretically possible for an attacker to use power consumption information to learn the value of all the bits of the secret key.*

In general, a sound attack may or may not be practical. Evaluating the practicality of a sound attack will usually require direct experimentation or a thorough simulation of a specific implementation. In this paper, I will confirm the soundness of two attacks, a first-order DPA attack and a second-order DPA attack. I will then examine the practicality of these attacks using an ST16 smartcard. These results likely represent the first documented analysis of an actual second-order DPA attack.

1.2 Power Leakage Model

For the attacks described in this paper, I assume that the processor will leak information about the Hamming weight of the data being processed. I also assume that processing data with higher Hamming weight will consume more power than processing data with lower Hamming weight and that this relationship is roughly linear. Such assumptions are not unreasonable since my research has confirmed that many present-day smartcard processors can exhibit precisely these characteristics.

Let the power consumption at a particular time j be represented by $P[j]$. The value of P[j] can be split into three parts. The first part represents the power contribution that varies with the Hamming weight of the data being processed. The second part represents a constant additive portion and the third part represents noise. This simple linear relationship for $P[j]$ can be written as

$$P[j] = \varepsilon \cdot d[j] + L + n \qquad (1)$$

where, $d[j]$ represents the Hamming weight of the intermediate data result at time j, ε represents the incremental amount of power for each extra '1' in the Hamming weight, L represents the additive constant portion of the total power, and n represents the noise. The noise n is assumed to have zero mean, thus when sufficient statistical averaging is used, this noise can usually be ignored.

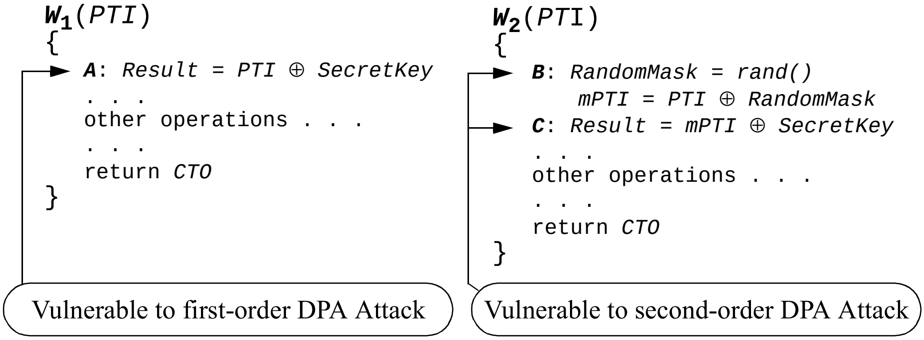
1.3 Power Attack Countermeasures

Goubin et al. [8] proposed a strategy, called the "duplication method", to protect the DES algorithm from first-order DPA attacks. Their countermeasure works by splitting secret data into two random halves and operating on each half separately. Such an approach causes the power consumption signals to be randomized, thus thwarting DPA. Similar techniques were also proposed to protect the advanced encryption standard algorithms from power attacks [9]. As a generalization, Chari et al. [10] suggested a countermeasure that splits the data into k shares. They proved that the amount of analysis needed to attack such a scheme increases exponentially with respect to k.

Secret splitting schemes protect against first-order DPA attacks, but they may leave an implementation susceptible to higher-order attacks. For some situations, this susceptibility might not be an issue because higher-order attacks are considered to be more difficult. For instance, in a recent paper Daemen et al. summarize that second-order DPA attacks require more complex analysis, increased memory and processing requirements, and an increased number of power consumption measurements [11]. One goal of this paper is to probe the complexity of a second-order DPA attack by investigating a specific example attack. Such research is necessary to ensure the design of secure countermeasures.

1.4 Example Data-Whitening Routines

To better understand the concept of a second-order DPA attack, it is useful to consider some simple examples. The pseudocode for algorithm segments, W_1 and W_2, are given in Fig. 1. These algorithms begin by combining the input *PTI* data with a secret key. This combining step is sometimes referred to as a "whitening process" and is used as a first step in some algorithms; i.e., a specific example is in the Twofish encryption algorithm [12]. The whitening of the input data is performed using the XOR operation. The W_1 algorithm immediately performs this XOR operation at line A. Unfortunately,



```
W1(PTI)
{
    A: Result = PTI ⊕ SecretKey
    . . .
    other operations . . .
    . . .
    return CTO
}
```

```
W2(PTI)
{
    B: RandomMask = rand()
       mPTI = PTI ⊕ RandomMask
    C: Result = mPTI ⊕ SecretKey
    . . .
    other operations . . .
    . . .
    return CTO
}
```

Vulnerable to first-order DPA Attack

Vulnerable to second-order DPA Attack

Fig. 1. Routines that Are Vulnerable to DPA Attacks
The routine on the left is vulnerable to a first-order DPA attack at line A. The routine on the right is safe from first-order attacks, but is vulnerable to second-order DPA. An attacker can mount a second-order DPA attack by using joint statistics on the power consumed when executing the operations at lines B and C.

the XOR operation at line A can leak information about the secret key. Thus, the W_1 algorithm is potentially susceptible to a first-order DPA attack.

In an attempt to avoid this DPA attack, the W_2 algorithm takes an indirect approach to the whitening operation. First, a random mask is generated at line B. Then, the XOR of the *PTI* data and the random mask are computed to produce intermediate result *mPTI*. Next, the XOR of *mPTI* and the secret key is computed at line C. The random mask is generated internally and is not observable to an attacker. Thus, when considered separately, the results of the operations at lines B and C leak only random information, and a DPA attack is prevented. However, when considered jointly, the operations at lines B and C are vulnerable to a second-order DPA attack.

2 Comparison of First and Second-Order DPA Attacks

The analysis in this section looks at specific attacks against the algorithms shown in Fig. 1. The attacks described here are proven to be sound. The specific steps for one possible DPA attack against the W_1 algorithm are now outlined in the following proposition.

Proposition 1. When the W_1 algorithm is implemented in an N-bit processor, where there is a linear relationship between the instantaneous power consumption and the Hamming weight of the data being processed, the following DPA attack is sound:

```
1. Repeat for i equal to 0 through $N-1$ {
2.     Repeat for $b = 0$ to 1 {
3.         Calculate the average power signal $A_b[j]$ by repeating the following: {
4.             Set the ith bit of the PTI input to b.
5.             Set the remaining PTI bits to random values.
6.             Collect the algorithm's power signal. } }
7.     Create the DPA bias signal $T[j] = A_0[j]$ - $A_1[j]$.
8.     $T[j]$ will have a positive bias spike when the ith secret key bit is a one, and
       will have a negative DPA bias spike when ith secret key bit is a zero. }
```

Proof. Let j^* be the sample time that corresponds to the time at which the result of line A in the W_1 routine is calculated. Also, let the power consumption at this time be represented by P. Thus, using the model of Equation (1), $P = d\varepsilon + L + n$, where d represents the Hamming weight of the variable *Result* at line A of W_1.

Denote the ith bits of the *SecretKey* and the *PTI* data as k_i and p_i, respectively. The expected value of the Hamming weight d depends on the values of k_i and p_i as follows:

$$\mathsf{E}\left[d \mid k_i \oplus p_i = 0\right] = \frac{N-1}{2} \qquad \mathsf{E}\left[d \mid k_i \oplus p_i = 1\right] = \frac{N+1}{2}$$

When $k_i = 0$, equations for $A_0[j^*]$ and $A_1[j^*]$ can be written in terms of the expected values of P

$$A_0[j^*] \approx \mathrm{E}[P \mid k_i = 0, p_i = 0] = \mathrm{E}[d\varepsilon + L + n \mid k_i = 0, p_i = 0] = \frac{N-1}{2}\varepsilon + L \quad (2)$$

$$A_1[j^*] \approx \mathrm{E}[P \mid k_i = 0, p_i = 1] = \mathrm{E}[d\varepsilon + L + n \mid k_i = 0, p_i = 1] = \frac{N+1}{2}\varepsilon + L \quad (3)$$

Taking the difference of Equations (2) and (3) yields

$$T[j^*] = A_0[j^*] - A_1[j^*] \approx -\varepsilon \quad \text{when } k_i = 0 \quad (4)$$

Similarly, when $k_i = 1$, equations for $A_0[j^*]$ and $A_1[j^*]$ can be written in terms of the expected values of power consumption P

$$A_0[j^*] \approx \mathrm{E}[P \mid k_i = 1, p_i = 0] = \mathrm{E}[d\varepsilon + L + n \mid k_i = 1, p_i = 0] = \frac{N+1}{2}\varepsilon + L \quad (5)$$

$$A_1[j^*] \approx \mathrm{E}[P \mid k_i = 1, p_i = 1] = \mathrm{E}[d\varepsilon + L + n \mid k_i = 1, p_i = 1] = \frac{N-1}{2}\varepsilon + L \quad (6)$$

Taking the difference of Equations (6) and (5) yields

$$T[j^*] = A_0[j^*] - A_1[j^*] \approx \varepsilon \quad \text{when } k_i = 1 \quad (7)$$

So, it is clear from Equations (4) and (7) that there should be a positive bias spike when $k_i = 1$ and a negative bias spike when $k_i = 0$. Thus, Proposition 1 is a sound DPA attack. □

2.1 Second-Order DPA attack

Now, consider the W_2 algorithm on the right side of Fig. 1. This algorithm is not vulnerable to the DPA attack of Proposition 1. Instead of directly calculating the XOR of the *SecretKey* and *PTI*, this algorithm first generates a random variable *RandomMask* to mask the value of *PTI*. The secret key is used at line *C*, but the Hamming weight of the result is random. Thus, the power consumption of the result at line *C* cannot be correlated to the values of the secret key or the *PTI* data. The W_2 algorithm seems secure against a first-order DPA attack, yet a second-order DPA attack is definitely possible.

Proposition 2. When the W_2 algorithm is implemented in an N-bit processor, where there is a linear relationship between the instantaneous power consumption and the Hamming weight of the data being processed, the following second-order DPA attack is sound:

```
1. Repeat for i equal to 0 through $N-1$ {
2.     Repeat for $b = 0$ to 1 {
3.         Calculate average statistic $\bar{S}_b = |P_B - P_C|$ by repeating the following:{
4.             Set the ith bit of the PTI input to b.
5.             Set the remaining PTI bits to random values.
6.             Collect the algorithm's instantaneous power consumption as lines B
                   and C. Call these values $P_B$ and $P_C$, respectively. } }
7.     Calculate the DPA bias statistic $T = \bar{S}_0 - \bar{S}_1$ .
8.     If $T > 0$ then the ith key bit is a one, otherwise it is a zero.
```

Proof. The power consumption at lines B and C of W_2, respectively P_B, P_C, can be modeled using the linear relationships:

$$P_B = d_B\varepsilon_B + L_B \text{ and } P_C = d_C\varepsilon_C + L_C \tag{8}$$

where, d_B represents the Hamming weight of the data *RandomMask* at line B, d_C represent the Hamming weight of the data *Result* at line C, ε_B and ε_C represent the extra amount of power for each '1' in the data at lines B and C, and L_B and L_C represent the constant portions of the total power at lines B and C. The noise contributions are ignored, since when averaging is used, these contributions will be removed. Also, to simplify the proof, I initially assume that

$$L_B = L_C \text{ and } \varepsilon_B = \varepsilon_C \tag{9}$$

My experimental results confirmed that the assumptions of equality in Equation (9) are true for the implementation I considered. However, in the general case these equalities may not hold. For this proof I will first consider the case where Equation (9) holds. Then, at the end of this proof I will comment on the more general case.

In the second-order DPA attack of Proposition 2, the value of $|P_B - P_C|$ is used as a statistic to determine the value of the ith bit of the key. The value of $|P_B - P_C|$ can be rewritten by using Equations (8) and (9),

$$|P_B - P_C| = \varepsilon|d_B - d_C|$$

where, $\varepsilon = \varepsilon_B = \varepsilon_C$. Now, refer back to the W_2 algorithm in Fig. 1. Let the ith bit of the variable *SecretKey* be k_i, the ith bit of the random variable *RandomMask* be r_i and the ith bit of *PTI* be p_i. Recall that the variables d_B and d_C are random variables corresponding to the Hamming weights of the N-bit data processed at lines B and C, respectively. The expected value of d_B is dependent on r_i and the expected value of d_C is dependent on the values of r_i, k_i and p_i

$$\begin{aligned} \mathsf{E}\left[d_B \mid r_i = 1\right] &= \mathsf{E}\left[d_C \mid r_i \oplus k_i \oplus p_i = 1\right] = (N+1)/2 \\ \mathsf{E}\left[d_B \mid r_i = 0\right] &= \mathsf{E}\left[d_C \mid r_i \oplus k_i \oplus p_i = 0\right] = (N-1)/2 \end{aligned} \tag{10}$$

Assuming that the variable *RandomMask* is uniformly distributed and using Equation (10), the values of $\bar{S}_0$ and $\bar{S}_1$ in the attack of Proposition 2 can now be calculated. Recall that when $\bar{S}_0$ is calculated, the ith bit of *PTI* is set to a zero, and that when $\bar{S}_1$ is calculated, the ith bit of *PTI* is set to a one. Thus, when $k_i = 0$, the value of $\bar{S}_0$ can be derived

$$\begin{aligned} \bar{S}_0 &= \frac{1}{2}\mathsf{E}\left[\varepsilon|d_B - d_C| \,\middle|\, r_i = k_i = p_i = 0\right] + \frac{1}{2}\mathsf{E}\left[\varepsilon|d_B - d_C| \,\middle|\, r_i = 1,\ k_i = p_i = 0\right] \\ &= 0 \end{aligned} \tag{11}$$

and the value of $\bar{S}_1$ can be derived

$$\begin{aligned}\bar{S}_1 &= \frac{1}{2}\mathrm{E}\left[\varepsilon\left|d_B - d_C\right|\middle| p_i = 1,\ r_i = k_i = 0\right] + \frac{1}{2}\mathrm{E}\left[\varepsilon\left|d_B - d_C\right|\middle| r_i = p_i = 1,\ k_i = 0\right] \\ &= \varepsilon\end{aligned} \tag{12}$$

The combination of Equations (11) and (12) yields

$$T = \bar{S}_0 - \bar{S}_1 = -\varepsilon$$

In the case where $k_i = 1$, the derivation of $\bar{S}_0$ and $\bar{S}_1$ is very similar except that the results are swapped, $\bar{S}_0 = \varepsilon$ and $\bar{S}_1 = 0$. Therefore, when $T < 0$, then $k_i = 0$, and when $T > 0$, then $k_i = 1$. Hence, the sign of T indicates the value of k_i and the attack in Proposition 2 is a sound second-order DPA attack. □

Remark. The proof of Proposition 2 is based on the assumption in Equation (9) that certain parameters are equal. When this equality assumption is not true, the situation can be handled through a process of normalization. Instead of calculating $\bar{S}_0$ and $\bar{S}_1$ by directly using P_B and P_C, normalized versions of P_B and P_C can be used. Normalized versions of P_B and P_C are calculated by subtracting the mean and dividing by the variance. For example, a normalized version of P_B is calculated as

$$\text{normalized } P_B = (P_B - \mathrm{E}[P_B])/\mathtt{var}[P_B \tag{13}$$

By using normalized values for P_B and P_C, the equality assumption of Equation (9) is effectively forced to be true, thus resulting in a sound attack.

3 Experimental Results

In this section I provide experimental results showing the practical aspects of the previously described DPA attacks. The EEPROM memory of an ST16 smartcard was programmed with versions of the W_1 and W_2 algorithms from Fig. 1. Then, the attacks from Propositions 1 and 2 were implemented and tested against this smartcard. Statistical results from measuring the smartcard's power consumption were collected and analyzed.

In a first-order DPA attack, knowledge of design information is not required. In a second-order DPA attack, however, knowledge of the algorithm code and the processor operation is much more important. Without such knowledge, attackers will not know which points in the power consumption signal are important. For example, in the W_2 algorithm these points correspond to the execution of lines B and C. Attackers that do not know which points to observe will need to resort to additional statistical analysis to find these points. Although possible, such an approach makes an attack more difficult, especially as the order of the DPA attack grows. To avoid these complications in my experiments, I assumed that the attacker knows exactly which points in the power trace to monitor.

With knowledge of which points in a power trace to monitor, it becomes easy to implement both DPA attacks. The hardware used for this experiment was simply a PC, a digital sampling oscilloscope, and a smartcard reader. The smartcard reader was physically modified to allow for easy power measurements. This modification entailed placing an 18 ohm resistor in series between the smartcard and reader's ground pins. Power consumption was monitored by sampling the voltage across this resistor. The smartcard was clocked at 3.57 Mhz and the power signal was asynchronously sampled by setting the oscilloscope's sampling rate to 1.0 Gsamples/second.

Implementors of smartcard systems often wonder how much sample data an attacker will need for a successful attack. I designed an experiment that looks at this question for my given ST16 implementations. In my experiments, DPA attacks were run and power signals were collected. As each power signal was collected, an updated value of *T* was calculated. The accuracy of my attacks increased as the number of power signals used to calculate *T* increased. Thus, as the number of power signals increased, the sign of *T* converged to be either positive or negative, depending on the value of the bit being attacked.

My experiments were run a number of times and typical graphs showing the convergence of *T* versus the number of power signals are given in Figs. 2 and 3. The results of the first-order DPA attack in Proposition 1 are shown in Fig. 2. The plots in this figure show the convergence of *T* for each bit of the byte being attacked. In this example, the byte being attacked is equal to 0x6B. Thus, for bit #0, *T* should converge to a positive value; for bit #1, *T* should converge to a positive value; for bit #2, *T* should converge to a negative value, etc. My experimental result confirmed that this attack is practical. Fewer than 50 power signals were needed for *T* to converge to the correct value for all bits. In fact, for most bits, *T* converged with much fewer than 50 signals.

My second experimental results, that confirm the attack of Proposition 2, are given in Fig. 3. Again, it is clear that this attack is practical. An interesting observation is that *T* converges at different rates for different bits in a byte. For some bits, *T* converged quickly; fewer than 50 power signals were needed. However, for other bits, *T* converged more slowly. For example, in Fig. 3, bit #5 requires about 2,500 power signals before *T* stabilizes to the correct sign. In general, the convergence of *T* in the second-order attack is slower and more erratic than in the first-order attack. Surprisingly, however, for some bits, *T* converges nearly as fast for both attacks.

It should be stressed, however, that these experimental results apply only to my specific ST16 implementation that was being tested. Implementors of other systems will need to test or simulate their own implementations to accurately assess any vulnerabilities.

4 Developing an Optimal Second-Order DPA Attack

The attack from Proposition 2 is based on the statistic $\bar{S} = E[|P_B - P_C|]$. The formula for statistic $\bar{S}$ was chosen using an ad hoc approach based on the linear model of the power consumption signal. Although, an attack using $\bar{S}$ was experimentally shown to be practical, statistics that use other combinations of P_B and P_C may lead to even better attacks. For example, Chari et al. [10] suggest an alternate statistic, based on multiplying P_B and P_C, rather than taking their difference. Finding the optimal statistic for a second-order DPA attack is the topic that will now be investigated.

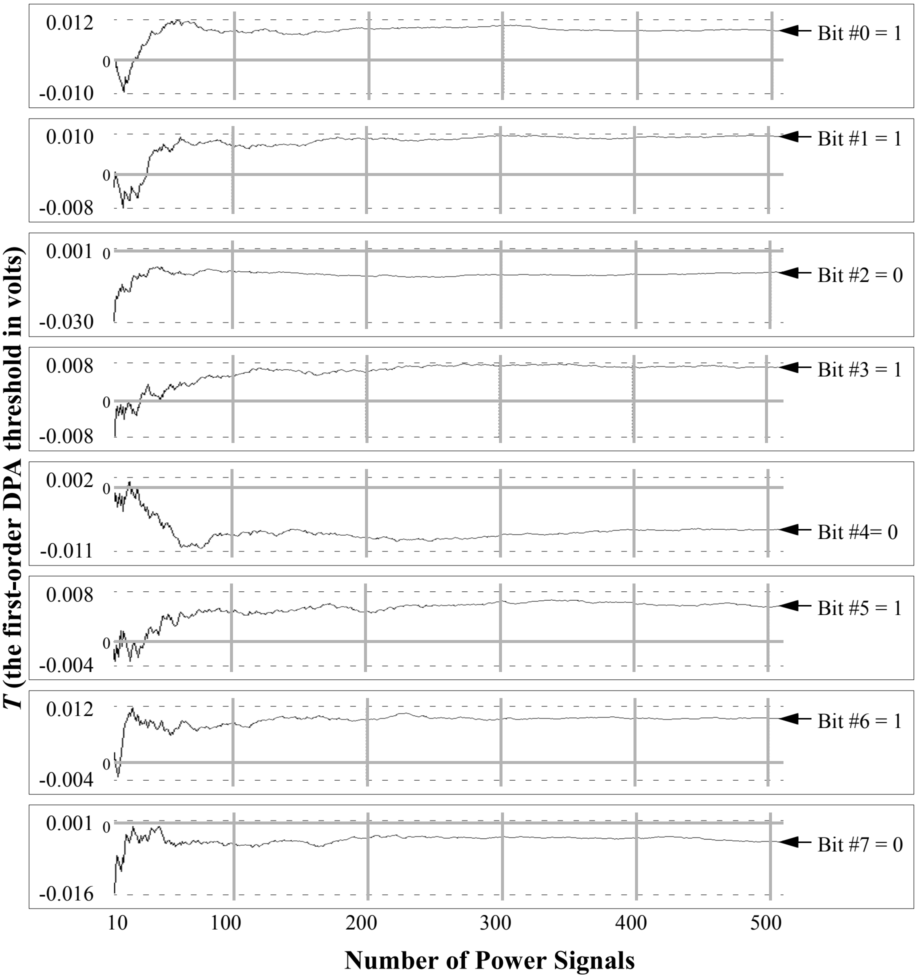


Fig. 2. First-Order DPA Threshold versus the Number of Power Signals
The above plot shows the convergence of *T* versus the number of power signals. The byte being attacked is equal to 0x6B and the resulting convergence plots for each bit of this byte are given above. The horizontal shaded lines denote the axis where *T* equals zero. A positive value for *T* indicates a bit is a one and a negative value indicates a bit is a zero. In all cases, *T* converges to the correct bit using fewer than 50 power signals.

A DPA attack against a secret bit of a key is a perfect example of a classic decision problem. Given noisy power consumption data, an attacker needs to decide whether a key bit is a zero or a one. An optimal decision is made when the probability of a wrong decision is minimized. Given a set of power consumption data, one can compute probabilities to determine the optimal decision. Let k_i, r_i and p_i be defined as before and let Ψ represent all of the observed power consumption data. During an attack, I assume that adversaries know p_i, so it is sufficient for them to determine $k_i \oplus p_i$. In an optimal

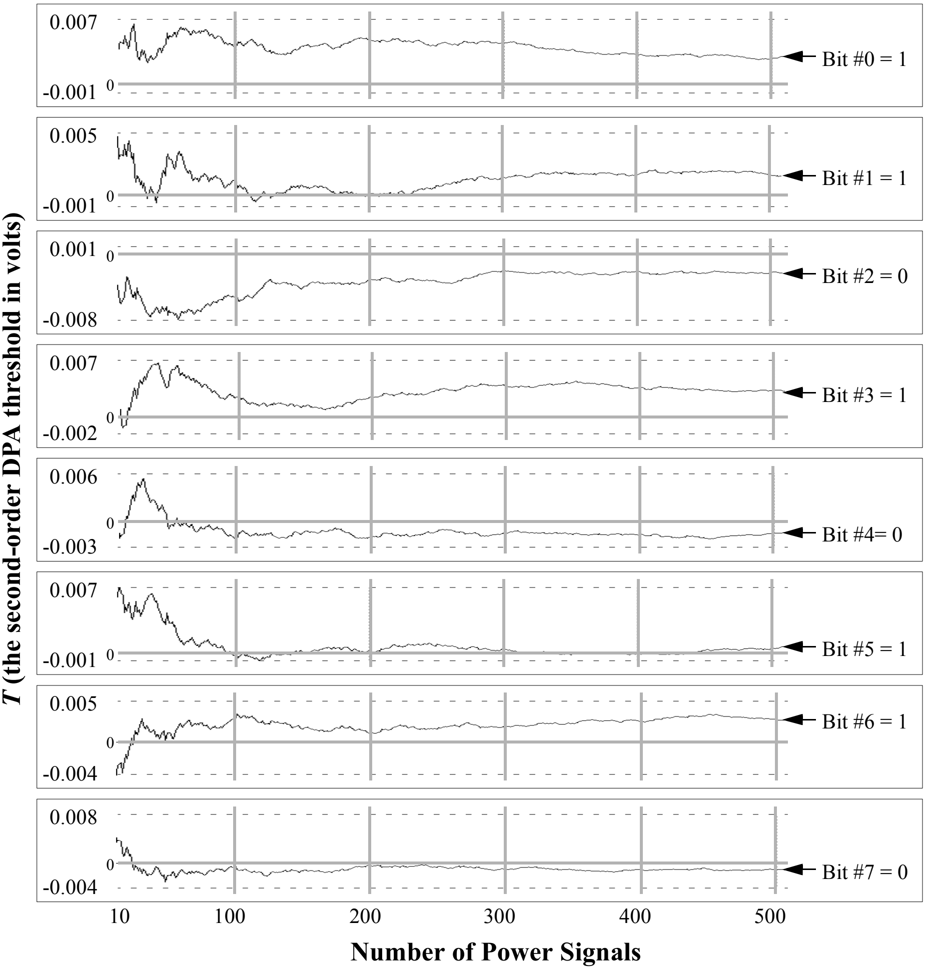


Fig. 3. Second-Order DPA Threshold versus the Number of Power Signals
The above plot shows the convergence of *T* versus the number of power signals. The byte being attacked is equal to 0x6B and the resulting convergence plots for each bit of this byte are given above. The horizontal shaded lines denote the axis where *T* equals zero. A positive value for *T* indicates a bit is a one and a negative value indicates a bit is a zero. In most cases *T* converges to the correct bit using fewer than 50 power signals. However, in the case of bit #5, *T* requires more than 2,500 power signals to converge.

attack, the attacker will choose the value for $k_i \oplus p_i$ that was more likely to have produced the observed data Ψ. Symbolically, the decision problem is reduced to solving the inequality

$$\Pr[\Psi|k_i \oplus p_i = 0] \lessgtr \Pr[\Psi|k_i \oplus p_i = 1] \tag{14}$$

The job of the attacker is to calculate both of the probabilities in Equation (14). The larger of the probabilities indicates the most likely value for $k_i \oplus p_i$. To begin calculating these probabilities, let the observed power consumption data Ψ be represented by N vectors, where each vector is the power consumption data from a single run of the algorithm. In a second-order DPA attack, the kth vector is composed of two elements (b_k, c_k), where b_k and c_k represent the instantaneous power consumptions for the two instructions being attacked. In the W_2 algorithm, b_k and c_k represent the normalized power consumption of the instructions at lines B and C, respectively.

The power consumption values b_k and c_k are random variables having probability density functions f_b and f_c, respectively. The distributions for these variables is assumed to be Guassian and the distributions are

$$f_b(b_k) \sim \mathrm{N}(0, \hat{\sigma}^2) \qquad f_c(c_k) \sim \mathrm{N}(0, \hat{\sigma}^2)$$

The probability distributions for b_k and c_k, conditioned on k_i, r_i and p_i, are also Guassian.

$$\begin{array}{ll} f_b(b_k | r_i = 0) \sim \mathrm{N}(-\frac{\varepsilon}{2}, \sigma^2) & f_b(b_k | r_i = 1) \sim \mathrm{N}(\frac{\varepsilon}{2}, \sigma^2) \\ f_c(c_k | r_i \oplus k_i \oplus p_i = 0) \sim \mathrm{N}(-\frac{\varepsilon}{2}, \sigma^2) & f_c(c_k | r_i \oplus k_i \oplus p_i = 1) \sim \mathrm{N}(\frac{\varepsilon}{2}, \sigma^2) \end{array} \qquad (15)$$

For shorthand, the results of Equation (15) can be written as

$$\begin{array}{ll} f_b^- = f_b(b_k | r_i = 0) & f_b^+ = f_b(b_k | r_i = 1) \\ f_c^- = f_c(c_k | r_i \oplus k_i \oplus p_i = 0) & f_c^+ = f_c(c_k | r_i \oplus k_i \oplus p_i = 1) \end{array}$$

Using this shorthand notation and the assumption that r_i is equally likely to be a one or a zero, the joint conditional probability distributions of b_k and c_k can be shown to be

$$\begin{aligned} f_{b,c}(b_k, c_k | k_i \oplus p_i = 0) &= \frac{1}{2} f_b^- f_c^- + \frac{1}{2} f_b^+ f_c^+ \\ f_{b,c}(b_k, c_k | k_i \oplus p_i = 1) &= \frac{1}{2} f_b^- f_c^+ + \frac{1}{2} f_b^+ f_c^- \end{aligned} \qquad (16)$$

The joint conditional probabilities of Equation (16) can now be used to solve the decision problem that was given in Equation (14).

Theorem 1. An optimal second-order DPA attack using N vectors (b_k, c_k), where each vector is assumed to be independent and b_k and c_k are assumed to be jointly normal random variables, reduces to the decision problem

$$\prod_{k=0}^{N-1} \cosh(b_k + c_k) \lesseqgtr \prod_{k=0}^{N-1} \cosh(b_k - c_k)$$

Proof. Using Equation (16) and the assumption that the (b_k, c_k) vectors are independent for different values of k, the probabilities in Equation (14) can be shown to be

$$\mathtt{Pr}[\Psi|k_i \oplus p_i = 0] = \prod_{k=0}^{N-1} f_{b,c}(b_k, c_k | k_i \oplus p_i = 0) = \frac{1}{2^N} \prod_{k=0}^{N-1} (f_b^- f_c^- + f_b^+ f_c^+)$$

$$\mathtt{Pr}[\Psi|k_i \oplus p_i = 1] = \prod_{k=0}^{N-1} f_{b,c}(b_k, c_k | k_i \oplus p_i = 1) = \frac{1}{2^N} \prod_{k=0}^{N-1} (f_b^- f_c^+ + f_b^+ f_c^-)$$

The decision problem is to solve an inequality, so terms appearing in both equations can be dropped. The decision problem can now be written as

$$\prod_{k=0}^{N-1} (f_b^- f_c^- + f_b^+ f_c^+) \lessgtr \prod_{k=0}^{N-1} (f_b^- f_c^+ + f_b^+ f_c^-) \tag{17}$$

Now, the appropriate Gaussian distribution functions can be substituted in for f_b^-, f_c^-, f_b^+ and f_c^+. The left side of Equation (17) can now be written as

$$\prod_{k=0}^{N-1}\left(\exp\left\{-\frac{1}{2}\left[\frac{\left(b_k+\frac{\varepsilon}{2}\right)^2+\left(c_k+\frac{\varepsilon}{2}\right)^2}{\sigma^2}\right]\right\}+\exp\left\{-\frac{1}{2}\left[\frac{\left(b_k-\frac{\varepsilon}{2}\right)^2+\left(c_k-\frac{\varepsilon}{2}\right)^2}{\sigma^2}\right]\right\}\right)$$

$$\prod_{k=0}^{N-1}\left(\exp\left\{-\frac{1}{2}\left[\frac{b_k^2+c_k^2+b_k\varepsilon+c_k\varepsilon+\frac{\varepsilon^2}{2}}{\sigma^2}\right]\right\}+\exp\left\{-\frac{1}{2}\left[\frac{b_k^2+c_k^2-b_k\varepsilon-c_k\varepsilon+\frac{\varepsilon^2}{2}}{\sigma^2}\right]\right\}\right.$$

and the right side of Equation (17) can be written as

$$\prod_{k=0}^{N-1}\left(\exp\left\{-\frac{1}{2}\left[\frac{\left(b_k+\frac{\varepsilon}{2}\right)^2+\left(c_k-\frac{\varepsilon}{2}\right)^2}{\sigma^2}\right]\right\}+\exp\left\{-\frac{1}{2}\left[\frac{\left(b_k-\frac{\varepsilon}{2}\right)^2+\left(c_k+\frac{\varepsilon}{2}\right)^2}{\sigma^2}\right]\right\}\right)$$

$$\prod_{k=0}^{N-1}\left(\exp\left\{-\frac{1}{2}\left[\frac{b_k^2+c_k^2+b_k\varepsilon-c_k\varepsilon+\frac{\varepsilon^2}{2}}{\sigma^2}\right]\right\}+\exp\left\{-\frac{1}{2}\left[\frac{b_k^2+c_k^2-b_k\varepsilon+c_k\varepsilon+\frac{\varepsilon^2}{2}}{\sigma^2}\right]\right\}\right.$$

Again, the terms that appear in both equations can be cancelled, resulting in the left side of Equation (17) being reduced to

$$\prod_{=0}^{\surd-1}\left(\exp\left\{-\frac{\varepsilon}{2\sigma^2}[b_j+c_j]\right\}+\exp\left\{\frac{\varepsilon}{2\sigma^2}[b_j+c_j]\right\}\right) = \prod_{j=0}^{N-1}\cosh\left[\frac{\varepsilon}{2\sigma^2}(b_j+c_j)\right]$$

and the right side of Equation (17) being reduced to

$$\prod_{=0}^{-1}\left(\exp\left\{-\frac{\varepsilon}{2\sigma^2}[b_k-c_k]\right\}+\exp\left\{\frac{\varepsilon}{2\sigma^2}[b_k-c_k]\right\}\right) = \prod_{k=0}^{N-1}\cosh\left[\frac{\varepsilon}{2\sigma^2}(b_k-c_k)\right]$$

Finally, the common factor of $\frac{\varepsilon}{2\sigma^2}$ can be removed, resulting in the decision problem given by Theorem 1. □

Remark. When $|b_k - c_k| \gg 1$, the ad hoc attack in Proposition 2 is a close approximation to the optimal decision problem. This can be seen since when $|b_k - c_k| \gg 1$,

$$\prod_{\iota=0}^{V-1}\cosh(b_k-c_k) \propto \sum_{k=0}^{N-1}|b_k-c_k|$$

Thus, the optimal attack statistic is approximately proportional to the ad hoc attack statistic. To verify this result, I repeated my previous experiments using the optimal attack. The results confirmed that the optimal and ad hoc second-order attacks are approximately equal.

5 Countermeasures to Higher-Order DPA Attacks

The long-term solution to these attacks it to develop hardware that does not leak secret information. Examples for potentially secure hardware have been reported by Moore et al. [13] and Kessel [14]. Statistical tests such as those suggested in [15] can be used to evaluate such hardware. Until such hardware is deployed, many of the same countermeasures that are effective against first-order DPA attacks may also help resist higher-order DPA attacks. Adding random time delays that are difficult for an attacker to remove is one such countermeasure. Also, keeping the implementation details secret can be very effective against higher-order attacks. Unlike first-order DPA, higher-order DPA is more complex if these details are not known to the attacker. Secret splitting schemes, such as the one proposed by Chari et al. [10], may also be effective.

6 Conclusions

Whichever countermeasures are chosen, designers will need to test their implementations for specific vulnerabilities. The theoretical analysis and the example of a practical second-order DPA attack provided in the paper will hopefully help future designers make their implementations more secure.

References

1. J. Kelsey, B. Schneier, D. Wagner, and C. Hall, “Side Channel Cryptanalysis of Product Ciphers,” in *Proceedings of ESORICS ‘98*, Springer-Verlag, September 1998, pp. 97-110.
2. Paul Kocher, Joshua Jaffe, and Benjamin Jun, “Differential Power Analysis,” in proceedings of *Advances in Cryptology–CRYPTO ‘99*, Springer-Verlag, 1999, pp. 388-397.
3. Thomas S. Messerges, Ezzy A. Dabbish, and Robert H. Sloan, “Investigations of Power Analysis Attacks on Smartcards,” *Proceedings of USENIX Workshop on Smartcard Technology*, May 1999, pp. 151-161.
4. Eli Biham and Adi Shamir, *“*Power Analysis of the Key Scheduling of the AES Candidates*,” Second Advanced Encryption Standard Candidate Conference*, March 1999, `http://www.nist.gov/aes`.
5. S. Chari, C. Jutla, J.R. Rao, and P. Rohatgi, *“*A Cautionary Note Regarding Evaluation of AES Candidates on Smart-Cards*,” Second Advanced Encryption Standard Candidate Conference*, http://www.nist.gov/aes, March 1999.
6. Thomas S. Messerges, Ezzy A. Dabbish, and Robert H. Sloan, “Power Analysis Attacks of Modular Exponentiation in Smartcards,” in proceedings of *Workshop on Cryptographic Hardware and Embedded Systems*, Springer-Verlag, August 1999, pp. 144-157.
7. Jean-Sébastien Coron, “Resistance Against Differential Power Analysis for Elliptic Curve Cryptosystems,” in proceedings of *Workshop on Cryptographic Hardware and Embedded Systems*, Springer-Verlag, August 1999, pp. 292-302.
8. Louis Goubin and Jacques Patarin, “DES and Differential Power Analysis – The Duplication Method,” in proceedings of *Workshop on Cryptographic Hardware and Embedded Systems*, Springer-Verlag, August 1999, pp. 158-172.
9. Thomas S. Messerges, “Securing the AES Finalists Against Power Analysis Attacks,” in proceedings of *Fast Software Encryption Workshop 2000*, Springer-Verlag, April 2000.
10. Suresh Chari, Charanjit S. Jutla, Josyula R. Rao and Pankaj J. Rohatgi, “Towards Sound Approaches to Counteract Power-Analysis Attacks,” in proceedings of *Advances in Cryptology–CRYPTO ‘99*, Springer-Verlag, 1999, pp. 398-412.
11. Joan Daemen, Michael Peeters and Gilles Van Assche, “Bitslice Ciphers and Power Analysis Attacks,” in proceedings of *Fast Software Encryption Workshop 2000*, Springer-Verlag, April 2000.
12. Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, Chris Hall and Niels Ferguson, *The Twofish Encryption Algorithm : A 128-Bit Block Cipher*, John Wiley & Sons, 1999, ISBN: 0471353817.
13. S. W. Moore, R. Anderson and M. Kuhn, “Self-timed Technology to Reduce Smartcard Fraud,” in proceedings of *ACiD-WG Workshop*, Grenoble, February 2000.
14. Joep Kessels, “Applying Asynchronous Circuits in Contactless Smartcards,” in proceedings of *ACiD-WG Workshop*, Grenoble, February 2000.
15. Jean-Sébastien Coron, Paul Kocher and David Naccache, “Statistics and Secret Leakage,” in proceedings of *Financial Cryptography*, Springer-Verlag, February 2000.