Chapter 14
Improved Techniques for Side-Channel Analysis

Pankaj Rohatgi

14.1 Introduction

Over the last several years, side-channel analysis has emerged as a major threat to securing sensitive information in hardware and systems. The list of side-channels that have been (re)discovered include timing [8] micro-architectural anomalies [1, 5, 12, 13], power consumption [9], electromagnetic emanations [2, 7, 14], optical [10, 11] and acoustic leakage [4]. These side-channels have been used to break implementations of all major cryptographic algorithms (such as DES, AES, RSA, Diffie-Hellman, Elliptic curves, COMP128, etc.) both in software and in hardware as well as for extracting information directly from peripherals. Concurrently a variety of side-channel analysis techniques have been developed to perform these attacks. These techniques include simple power/EM analysis (SPA/SEMA), differential power/EM analysis (DPA/DEMA), higher-order DPA/DEMA, inferential power analysis (IPA), partitioning attacks, collision attacks, hidden Markov model, etc.

In fact, side-channel analysis is so powerful that most attacks succeed, in practice, using only a fraction of the information present within the side-channel(s)! Typically, these techniques do not analyze the characteristics of the noise present within the side-channel signals, but try to remove it by averaging over a large number of samples. Related leakages that occur at different times in a side-channel trace are not combined to extract more information, and leakages from multiple side-channels are rarely combined. Therefore, if such techniques fail to break an implementation using a small number of side-channel signals, it cannot be assumed that the implementation is immune to side-channel attacks involving a limited number of side-channel traces. This question is particularly important to vendors, since there are several system-level side-channel countermeasures [9] based on nonlinear key updates that rely on the assumption that an adversary cannot extract the key from a single (or few) side-channel trace(s). This question is also pertinent to

IBM T. J. Watson Research Center
e-mail: rohatgi@us.ibm.com

implementations of stream ciphers such as RC4 that have a rapidly (and nonlinearly) evolving internal secret state, where a side-channel attack must be able to recover the state before it gets changed.

Answering such questions related to the fundamental capabilities and limits of side-channel attacks requires a deeper understanding of side-channel leakages from a device and an information-theoretic analysis of the optimal side-channel attacks that are possible against it. In this chapter, we describe the theoretical foundations for such an analysis by presenting a leakage model for CMOS devices and the maximum likelihood principle as the information theoretic basis for determining the optimal attacks and limits of side-channel analysis. We introduce the multivariate Gaussian noise assumption that makes it practical to apply the maximum likelihood principle to side-channel analysis. We then describe several applications of this approach. The first application, *template attacks*, shows how implementations of stream ciphers such as RC4 that are immune to simple and differential side-channel attacks could be broken using a single side-channel trace. Since this classical template attack has several practical shortcomings we also describe single-bit template attacks that may be suboptimal but much more practical. We then describe other applications of the maximum likelihood approach, such as an improved metric for DPA/DEMA attacks, the design and analysis of attacks involving multiple side-channels, and for information leakage assessment.

14.2 CMOS Devices: Side-Channel Leakage Perspective

Side-channels such as power and EM from a CMOS device are directly attributable to the currents flowing within the device as it operates. The two basic types of current flows include *intentional* flows, which are currents that flow in accordance to the circuit design as it performs the computation, and *leakage* currents, which are a property of the technology used to fabricate the device. In addition, there is nonlinear electromagnetic coupling between the different currents flowing within the device which causes amplitude and angle modulation that in turn gives rise to several EM side-channels. In addition there are variations within the different current flows due to thermal noise.

14.2.1 Intentional Current Flows

In CMOS devices, all data processing is typically controlled by a “square-wave”-shaped clock. From a logical perspective, each clock edge causes the device to perform an *elementary operation* resulting in a change in the state of the device. From a physical perspective, the clock edge triggers a state-dependent sequence of switching events that result in current flows within the device. These events are transient and a steady state is achieved before the next clock edge. At any clock cycle, the events and resulting currents are dependent on only a small number of bits of the logic state of the device and not its entire state. These bits are termed *relevant bits* and consist of the bits of the state that change as well as the bits that influence the bits of the state that get changed. The set of *relevant bits* during a clock cycle constitute the *relevant state* of the device at that clock cycle.

14.2.2 Leakage Current Flows

In an ideal CMOS device, currents only flow when there is switching activity. However, due to shrinking feature sizes and usage of stressed silicon, there is a significant amount of current due to *leakage* even within the *inactive* parts of the circuit. The net leakage current within the circuit depends purely on the technology used and the size of the circuit. For our purposes we can approximate leakage current within a CMOS device as a constant plus a small Gaussian noise term that is uncorrelated to the activity occurring within the active part of the circuit.

14.2.3 Information Leakage in Power and EM Side-Channels

The power side-channel can be viewed as an aggregate measure of all the currents flowing within the device. Of these currents, only the intentional currents can provide information about the *relevant state* of the device. However, due to aggregation of currents and noise and due to impedances within the circuit and power grid, the influence of weak individual intentional currents on the power side-channel can be quite small. For understanding information leakage from EM, one only needs to consider coupling effects that involve at least one *intentional* current. Even a single EM sensor can pick up multiple and distinct mixtures of coupling effects over the entire EM spectrum.

As a very good first approximation, both power and EM side-channel emanations during a clock cycle carry information *only* about the relevant state of the device during the clock cycle and not the other parts of the device state.

This is strongly supported by the experimental results which show that algorithmic bits are significantly correlated to the power/EM signals *only* during the clock cycles where the bits are actively involved in a computation. While an algorithmic bit may leak to different degrees in different power and EM channels at different parts of the computation, it never leaks when it is inactive. For example, Figure 14.1 shows a power trace (in gray) overlaid with the contribution of a particular bit to power trace (in black). This bit only impacts the power trace during some clock cycles where it is part of the relevant state and not during all clock cycles. Figure 14.2 which plots the leakage of the same bit in different power/EM side-channels shows that the extent of leakage is different for different side-channels.

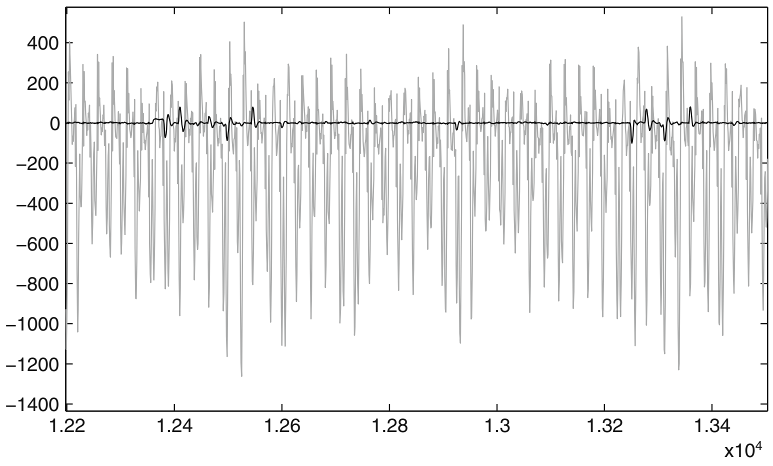

Fig. 14.1 Power side-channel (gray) overlayed with contribution from single relevant bit (black).

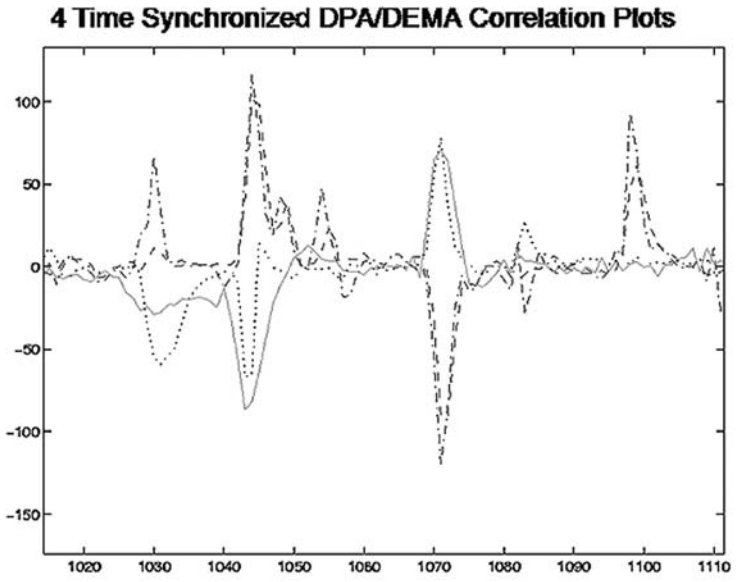

Fig. 14.2 DPA and three DEMA correlation curves (aligned).

14.3 Characterizing Side-Channel Leakage Using Maximum Likelihood

14.3.1 Adversarial Model

Given the side-channel leakage model above, it becomes natural to formulate side-channel attacks in terms of how successful an adversary can be in obtaining information about the *relevant state* using side-channels. For example, an adversary may be interested in the LSB of the data bus during a LOAD instruction or he may be interested in finding out the address of the data being loaded.

In general, the adversary would like to use side-channels to extract information about the relevant state of a device when it is performing an elementary operation given some prior knowledge about the relevant state. This is a classical inferencing problem, but for simplicity, we can assume that the adversary is attempting to find information about parts of the relevant state that are completely unknown. In this case the problem is naturally formulated as a hypothesis testing problem as follows:

The adversarial model consists of two phases. The first phase, known as the *profiling phase*, is a training phase for the adversary. He is given a *training device* identical to a *target device*, an elementary operation, k distinct probability distributions $B_1,\dots,B_k$ on the relevant states from which the elementary operation can be invoked and a set of sensors for monitoring side-channel signals.

The adversary can invoke the elementary operation, on the training device, starting from any relevant state. It is expected that adversary uses this phase to prepare an attack.

In the second phase, known as *the hypothesis testing phase*, the adversary is given a *target device* and the same set of sensors. He is allowed to make a *bounded number L* of invocations to the same elementary operation on the target device starting from a relevant state that is drawn *independently* for each invocation according to exactly one of the k distributions $B_1,\dots,B_k$. The choice of distribution is completely unknown to the adversary (i.e., a priori, each distribution is equally likely to be chosen with probability $\frac{1}{k}$) and his task is to use the signals on the sensors to select the correct hypothesis $(H_1,\dots,H_k)$ for the distribution being $B_1,\dots,B_k$. The utility of the side-channels to extract this information can then be measured in terms of the success probability achieved by the adversary as a function of the number of invocations L.

14.3.2 Maximum Likelihood and Best Attack Strategy

Assume that an adversary acquires L statistically independent sets of sensor signals $\mathbf{O}_i, i = 1,\dots,L$. These L sets of signals may correspond to L invocations of an operation on the target device. Also assume that there are K equally likely hypotheses H_k, $k = 1,\dots,K$, on the origin of these signals. Let $p(\mathbf{O}|H)$ be the probability distribution of the sensor signals under the hypothesis H. Under these assumptions, the *maximum likelihood hypothesis* test [15] is optimal and it decides in favor of the hypothesis H_k, if for all j, where $1 \leq j \leq K$

$$\prod_{i=1}^{L} p(\mathbf{O}_i|H_k) \geq \prod_{i=1}^{L} p(\mathbf{O}_i|H_j) \tag{14.1}$$

i.e., H_k is the hypothesis under which the actual observations have the highest probability of occurring.

While the maximum likelihood test is optimal, it is usually impractical as an exact characterization of the probability distribution of the sensor signals $\mathbf{O}$ may be infeasible. Such a characterization has to capture the nature of each of the sensor signals and the dependencies among them. This could further be complicated by the fact that, in addition to the thermal noise, the sensor signals could also display additional structure due to the interplay between properties of the device and those of the distributions of the relevant states.

It turns out that in practice one can obtain near-optimal results by making the right assumptions about the sensor signals. Such assumptions greatly simplify the task of hypothesis testing by requiring only a partial characterization of sensor signals.

14.3.3 Gaussian Assumption

One such widely applicable assumption is the *Gaussian assumption* which states that under the hypothesis H, the sensor signal $\mathbf{O}$ has a multivariate Gaussian distribution with mean μ_H and a covariance matrix Σ_H [15]. A multivariate Gaussian distribution $p(\cdot|H)$ has the following form:

$$p(\mathbf{o}|H) = \frac{1}{\sqrt{(2\pi)^n|\Sigma_H|}} \exp(-\frac{1}{2}(\mathbf{o}-\mu_H)^T \Sigma_H^{-1}(\mathbf{o}-\mu_H)), \quad \mathbf{o} \in \mathscr{R}^n \tag{14.2}$$

where $|\Sigma_H|$ denotes the determinant of Σ_H and Σ_H^{-1} denotes the inverse of Σ_H.

The Gaussian assumption holds for a large number of devices and hypotheses encountered in the practice. It can be shown that under the Gaussian assumption, the maximum likelihood hypothesis testing for a single observation $\mathbf{O}$ and two equally likely hypothesis H_0 and H_1[1] simplifies to the following comparison:

$$(\mathbf{O}-\mu_{H_0})^T \Sigma_{H_0}^{-1}(\mathbf{O}-\mu_{H_0}) - (\mathbf{O}-\mu_{H_1})^T \Sigma_{H_1}^{-1}(\mathbf{O}-\mu_{H_1}) \geq \ln(|\Sigma_{H_1}|) - \ln(|\Sigma_{H_0}|) \tag{14.3}$$

where a decision is made in favor of H_1 if the above comparison is true, and in favor of H_0 otherwise.

[1] Generalizations to multiple observations and more than two hypotheses are straightforward.

In signal processing it is common to treat the observed trace $\mathbf{O}$ as consisting of a mean signal component that depends purely on the operation being performed on the device and is fixed across multiple invocations and a noise component which can differ on each invocation. Noise is best modeled as a random sample drawn from a noise probability distribution having a mean of zero. In the equations above, if hypothesis H was correct then the mean signal component would be μ_H and the noise component in each sample, i.e., $\mathbf{O} - \mu_H$, would also have a multivariate Gaussian distribution with mean 0 and the same covariance matrix Σ_H. Thus the Gaussian assumption made here in the context of the signal characterization is often alternatively referred to as the Gaussian noise assumption.

In many cases of practical interest, noise in the sensor signals does not depend on the hypothesis, that is, $\Sigma_{H_0} = \Sigma_{H_1} = \Sigma_N$. In such cases, the following well-known result from statistics gives the probability of error in maximum likelihood hypothesis testing [15]:

Fact 1 *For equally likely binary hypotheses, the probability of error in the maximum likelihood testing is given by*

$$P_\varepsilon = \frac{1}{2}\operatorname{erfc}\left(\frac{\Delta}{2\sqrt{2}}\right) \tag{14.4}$$

where $\Delta^2 = (\mu_{H_1} - \mu_{H_0})^T \Sigma_N^{-1} (\mu_{H_1} - \mu_{H_0})$ *and* $\operatorname{erfc}(x) = 1 - \operatorname{erf}(x)$, *where*

$$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$$

is the error function. Note that Δ^2 *has a nice interpretation as the optimal signal-to-noise ratio that an adversary can achieve under the Gaussian assumption.*

In the rest of this chapter we will describe multiple applications of this characterization of side-channels and their leakage.

14.4 Template Attacks

The motivation behind the development of template attacks was that in several instances only a single (or a few) side-channel sample is available for carrying out an attack against a device. This situation arises naturally in the case of stream ciphers where the internal secret state keeps changing as the key stream is generated and in protocols where ephemeral keys are used. In addition, there are some system-level countermeasures that try to limit side-channel exposure by limiting the use of a particular key [9]. In such cases, the implementations can be easily made to be secure against traditional simple/differential attacks since typically the differences between signal levels with different keys/data is usually lower than the level of noise. This noise cannot be eliminated by averaging since there is only one or a very limited number of traces available.

For attacking such implementations we convert the model described in the last section into an attack technique called template attacks. If an adversary had infinite resources, the template attack would work using this basic principle: Suppose there is a crypto implementation on many "identical" devices and the adversary has access to one such device on which he can perform experiments and he is also given a single (or few) side-channel sample(s) S from a target device with an unknown key. The adversary uses the test device to build signal/noise models or *templates* for side-channel signals produced by the test device for *all* possible values of the key and uses the maximum likelihood to determine which key is used in the target sample.

Clearly, since key sizes are large, it is infeasible to build templates for all possible keys. Practical template attacks have to meld this basic attack principle with the details of the cryptographic algorithm being attacked. Typically this is done in an iterative fashion, where at each stage, the adversary starts with a small candidate set of prefixes for the key and ends with another small candidate set of larger-sized prefixes of the key. At the end of this process, the adversary has a limited set of complete keys that he can exhaustively test. In the beginning the candidate set is empty. At each step, the adversary uses the test device to identify a small sub-section of the sample S that depends only on a few unknown key bits. By experimenting with the test device, he builds signal templates corresponding to his set of candidate key prefixes extended by all possible value of the unknown key bits. The templates consist of the mean signal and (multivariate Gaussian) noise probability distributions for each of these extended prefixes of the key. He then compares these templates with the signal S and uses the maximum likelihood principle to retain only a small set of those prefixes that match S the best. Thus template attacks essentially use an extend-and-prune strategy directed by the single sample S to be attacked: the adversary extends candidate key prefixes by all possible values of a limited number of unknown key bits, builds templates, and uses template classification to prune the set of choices for these larger key prefixes. The success of this approach depends on the effectiveness of the pruning strategy in controlling the combinatorial explosion that occurs during the extension process.

Template attacks are particularly effective on implementations of cryptographic algorithms due to their *contamination* and *diffusion* properties. Contamination refers to key-dependent leakages which can be observed over multiple cycles in a section of computation. Additionally, other variables affected by the key, such as key-dependent table indices and values, cause further contamination at other cycles. The extent of contamination controls the success of the pruning of the fresh key bits introduced in the expansion phase. However, it is to be expected that if two keys are almost the same, that even with the effects of contamination, pruning at this stage may not be able to eliminate one of them. Diffusion is the well-known cryptographic property wherein small differences in key bits are increasingly magnified in subsequent portions of the computation. Even if certain candidates for key prefixes were not eliminated by contamination effects, diffusion will ensure that the wrong key prefixes get pruned rapidly at later stages.

We now provide the details on how such attacks could be carried out in practice by means of an example.

14.4.1 Classical Template Attacks: The Case of RC4

Consider an implementation of the stream cipher RC4. While there are cryptanalytic results on RC4 based on minor statistical weaknesses, none of these are useful for side-channel attacks. A well-designed system, RC4 implementations are also quite easy to secure against SPA- and DPA-style attacks. This is because initializing the 256-byte internal state of RC4 with a secret key is simple enough to be implemented using low leakage instructions in a key-independent manner. This makes SPA unlikely. After initialization, the only secret is the internal state. However, this secret state evolves very rapidly as the cipher outputs more bytes. This rapidly evolving secret state is outside the control of the adversary. This provides inherent immunity against statistical attacks such as DPA, since the adversary cannot freeze the active part of the state to collect multiple samples to eliminate the noise. For RC4, the best that an adversary can hope for is to obtain a *single* sample of the side-channel leakage during the key initialization phase and attempt to recover the key from that single sample.

We now describe how template attacks apply against RC4's state initialization routine. RC4 operates on a 256-byte state table T to generate a pseudo-random stream of bytes that is then XORed with the plaintext. Table T is initially fixed, and in the state initialization routine, a variable length key (1 to 256 bytes) is used to update T using the pseudo code below:

```
i1 = 0
i2 = 0
for ctr = 0 to 255
  i2 = (key[i1] + T[ctr] + i2) mod 256
  swap_byte(T[ctr], T[i2]);
  i1 = (i1 + 1) mod (key_data_len)
endfor
```

A portion of the corresponding power side-channel signal (plotted in gray) and the sample noise (plotted in black) for the first six iterations of the loop is shown in Figure 14.3.

First it needs to be verified that simple side-channel analysis techniques will not work on this implementation. This can be easily seen in Figure 14.4 which plots the noise level for the first six iterations in a power sample in gray and plots in black the difference between the signals for two different keys A and B that differ only in the first byte. The figure clearly shows that the level of noise in the first iteration (time 0 to 20 μs) far exceeds the differences between the signals for keys A and B in that iteration; so SPA will not have been able to determine which key byte was used in the first iteration. In fact, in [6], it was stated that averages of several tens of samples would be needed to reduce the level of noise below the signal differences.

RC4 is, however, an ideal candidate for a signal classification-based attack. Notice from the code snippet above that the key byte used in each iteration influences the computation (and is part of the relevant state) multiple times within a loop. For example, loading of the key byte, the computation of index `i2`, and the use of `i2`

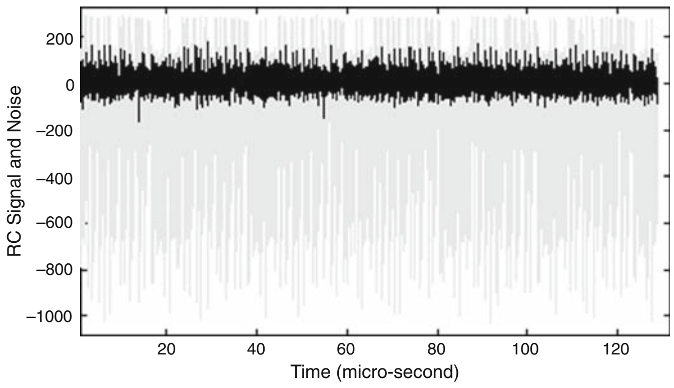

Fig. 14.3 Power signal (gray) and noise (black) during first six iterations of RC4 state initialization loop.

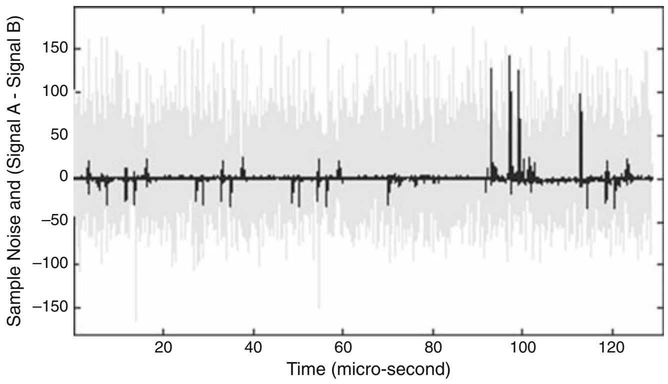

Fig. 14.4 Sample noise (gray) vs. signal differences (black) between keys A and B in first six rounds.

in swapping the bytes of the state table T all contaminate the side-channel at different cycles in the iteration. Thus RC4 demonstrates good *contamination* for the individual key bytes. Further, the use of `i2` and the state in subsequent iterations, and the fact that RC4 is a well-designed stream cipher, quickly propagates small key differences to cause diffusion. This analysis is borne out in practice as is shown in Figure 14.5 which plots the signal for the first six iterations for the key A in gray and the difference between the signals for key A and key B in black. Keys A and B differ only in the first byte and a small difference signal is clearly visible in the figure in the first iteration (0 to 20 μs). The important point to note is that even though the magnitude of the difference signal is small in the first iteration, significant differences appear at many different places in the first iteration, which indicates good contamination. The next point to note is that by the time the fifth iteration is reached, the difference signal has become quite large, indicating good diffusion.

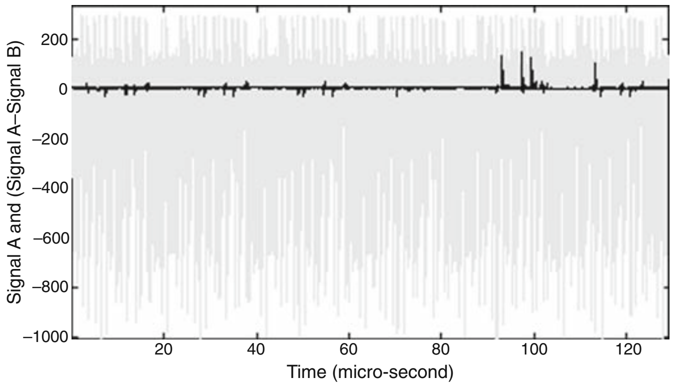

Fig. 14.5 Signal for key A (gray) vs. signal differences (black) between keys A and B in first six rounds.

The template attacks on this RC4 implementation works by building templates for the signal and noise for around 42 sample points in each iteration of RC4 state initialization routine that takes an input a fresh (unknown) key byte. These are the points where significant differences arise for different keys as shown in Figure 14.5.

A first attempt to use a statistical model where the noise at these 42 points was treated independently (i.e., Σ_H is a diagonal variance matrix) to classify the (unknown) key byte yielded poor results as shown in Table 14.1. The table here shows the classification rates assuming there were only five possible values of the key byte. Here the classification errors were as much as 35% for pairs of keys with few bit differences. Somewhat more encouraging was the fact that even this limited statistical model was fairly good (100% successful) at distinguishing between key byes that were very different.

Next the full multivariate noise approach was applied. In the experiment, there were 10 choices for the first key byte, as shown in column 1 of Table 14.2. They are carefully chosen to be very close and yielded poor results with the univariate statistics. For each key byte, 2000 side-channel signals were collected and analyzed at the same 42 points in time. The mean of the 2000 samples was used as the average signal for that key (μ_H) and the covariance matrix (Σ_H) was also computed from these signals. To obtain statistics on how well this approach works, the templates were used to classify tens of thousands of samples drawn using one of the

Table 14.1 Classification probability of five competing hypotheses using univariate statistics. Entry (i,j) is probability of classifying samples with key i as one with key j.

Key byte	1111 1110	1110 1110	1101 1110	1011 1110	0001 0000
1111 1110	0.86	0.04	0.07	0.03	0.00
1110 1110	0.06	0.65	0.10	0.20	0.00
1101 1110	0.08	0.16	0.68	0.09	0.00
1011 1110	0.10	0.11	0.08	0.71	0.00
0001 0000	0.00	0.00	0.00	0.00	1.00

Table 14.2 Percentage of samples for which the correct hypothesis is retained under different ball sizes with 10 competing hypotheses.

Key byte	Ball size $c = 1$	Ball size $c = e^6$	Ball size $c = e^{12}$	Ball size $c = e^{24}$
1111 1110	98.62	99.46	99.88	99.94
1110 1110	98.34	99.82	99.88	99.88
1101 1110	99.16	100.00	100.00	100.00
1011 1110	98.14	99.52	99.82	100.00
0111 1110	99.58	99.76	99.89	99.94
1111 1101	99.70	99.94	99.94	99.94
1111 1011	99.64	99.82	99.82	99.89
1111 0111	100.00	100.00	100.00	100.00
1110 1101	99.76	99.82	99.88	99.88
1110 1011	99.94	100.00	100.00	100.00
Average	99.29	99.81	99.91	99.95

10 choices as the first key byte. Column 2 in Table 14.2 summarizes the results of the classification experiments for this set of 10 key choices. Since the values were carefully chosen to reflect the worst case, these results can be extrapolated to the case of 256 different values of the key byte. Column 2 in Table 14.4 is an extrapolation of our results for the case of 256 different templates by making pessimistic assumptions about the number of "close" keys. In practice the actual results should be much better.

To iteratively apply the approach a first heuristic would be to retain only the most likely hypothesis, i.e., with highest likelihood probability. Even with such a drastic pruning approach, average classification success probability is 99.3% with these 10 hypotheses and worst-case probability was 98.1%. Detailed results are described in column 2 of Table 14.2. One gets reasonable results even if we use this extreme pruning strategy in each iteration of the extend-and-prune approach. Extrapolating, as shown in column 2 of Table 14.4, one can expect the average error probability of the closest hypothesis approach to be about 5–6% when we consider all 256 possible values, since pessimistically one expects around 50–60 keys to be "close" to any key. By bounding the error probability over many iterations by the sum of error in each iteration, it can be seen that when the number of key bytes is small this can be used to extract all key bytes. For example, one can do better than 50% for about eight bytes of key material.

With a little more effort, much better results can be obtained by using a *ball approach* to pruning. In this approach, a constant c is chosen and if the best hypotheses has probability P then all hypotheses that have probability P/c are retained. This approach is analogous to retaining all hypotheses which are a certain radius away from the top hypothesis and hence the term *ball approach*. The columns 2, 3, 4, and 5 of Table 14.2 showing success probability of retaining the correct hypothesis for balls with different values of c, with column 2 corresponding to $c = 1$ and retaining only the most likely hypothesis. When $c = e^6$, the average success probability has improved to better than 99.8% with the worst-case probability being 99.5%. As shown in Table 14.3 the average number of hypotheses that we retain is still close to 1

Table 14.3 Expected number of hypotheses retained under different ball sizes for 10 competing hypothesis.

	Ball size $c = 1$	Ball size $c = e^6$	Ball size $c = e^{12}$	Ball Size $c = e^{24}$
	1	1.041	1.158	1.842

Table 14.4 Extrapolated results for 256 competing hypotheses.

	Ball size $c = 1$	Ball size $c = e^6$	Ball size $c = e^{12}$	Ball size $c = e^{24}$
Success prob.	95.02	98.67	99.37	99.65
Retained hypotheses	1	1.29	2.11	6.89

for balls of size e^6 and e^{12}. Again, using an estimate of about $50-60$ close keys, we can extrapolate these results as done in Table 14.4. For example, choosing the ball size e^6, with good probability we expect to retain at most 1.5 hypotheses on the average, yet we are guaranteed to retain the correct hypothesis with probability at least 98.67%. Using this approach independently in each iteration, we can correctly classify keys of size n bytes with expected probability around (100–1.33n)% and the number of remaining hypotheses would grow no more than $(1.5)^k$, which is substantially better than 2^{8k} (the entropy of the key).

14.4.2 Single-Bit Templates and Applications

The classical template attack described above suffers from several drawbacks. One major drawback is that the methodology requires an iterative approach that attempts to make test device's computation identical to that of the target device. This makes the attack tedious, iterative, and online. For example, in the RC4 case 256 templates for each unknown byte have to be constructed and templates for later bytes cannot be constructed until the earlier bytes have been attacked. Another drawback is that classical template attacks cannot handle randomized implementations, since the attacker cannot force a test device to produce the same randomness as the target device.

Single-bit template attacks are an attempt to get past these limitations at the cost of reduced classification accuracy. These attacks are based on an empirical observation that after a successful DPA attack on an algorithmic bit b, the DPA peaks themselves could be used to create binary templates to extract the bit b directly from any signal! This means that once a particular implementation/device has been attacked using DPA (say using a test card) one can predict the internal bits occurring within a single trace from an identical implementation/device.

We illustrate the attack by means of an example. Consider an unprotected implementation of DES on smartcard A. Consider the 32 s-box output bits of the DES computation in round one. For the unprotected DES implementation, one can easily perform DPA for each of the 32 output bits. Correspondingly, one can build a pair of templates for each output bit corresponding to the bit being equal to 0 and 1,

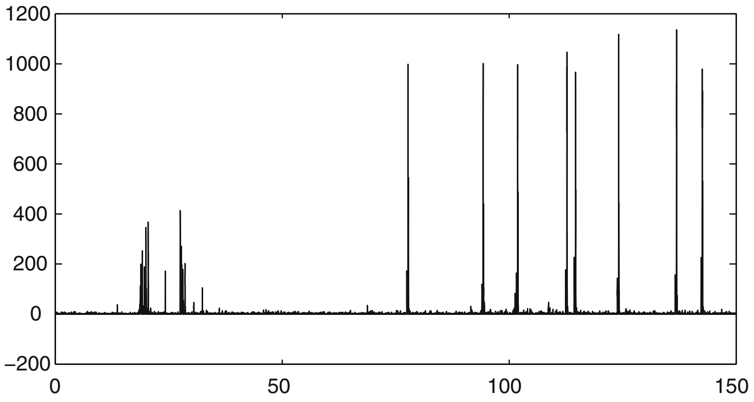

Fig. 14.6 Improved DPA metric of s-box 1, bit 0 of the test device. Time in μs.

respectively. In order to build these templates, a DPA attack was performed on each output bit using a DPA metric that we will be describing in Section 14.5.

Figure 14.6 displays the DPA metric of s-box 1, bit 0. The figure reveals several points in time that clearly correlate with the selected s-box output bit. In the experiments, the 50 highest peaks from this DPA metric were selected to be the points that were incorporated into the pair of templates (bit = 0 and bit = 1) for that bit. Templates were built for each s-box output bit using a single set of 1400 side-channel samples.

To estimate classification success rate, a set of 100 fresh random side-channel samples were collected from the same device and all 32 s-box output bits were classified using the templates developed earlier. The classification success rates $\eta_{S_i b_j}$ for the ith s-box and jth bit, $1 \leq i \leq 8$ and $0 \leq j \leq 3$, together with the corresponding entropy loss are shown in Table 14.5. The classification success rates ranged from 0.72 to 1.00; in the worst case s-box 3, bit 3 and s-box 6, bit 0 were predicted correctly for only 72 of the 100 samples. From these results, the probability that the entire 32-bit output of all s-boxes is classified correctly is $\prod_{i=1}^{8} \prod_{j=0}^{3} \eta_{S_i b_j} = 0.0154$ which although small is still 66 million times higher than a random guess.

These results can also be viewed in terms of entropy loss. For a particular bit, if the classification success rate is p, then its corresponding entropy loss is given by $1 + (1-p)\log_2(1-p) + p\log_2(p)$. To compute the entropy loss for multiple bits we

Table 14.5 s-Box output bit classification success rates and entropy loss.

	s-box 1	**s-box 2**	**s-box 3**	**s-box 4**	**s-box 5**	**s-box 6**	**s-box 7**	**s-box 8**
bit 0	1.00	0.91	0.88	0.93	0.77	0.72	0.80	0.84
bit 1	0.98	0.88	0.92	0.94	1.00	0.92	0.97	0.77
bit 2	0.75	0.89	0.99	0.92	0.95	0.83	0.90	0.79
bit 3	0.90	0.91	0.72	0.85	0.83	0.86	1.00	0.89
Entropy loss	2.57	2.10	2.13	2.30	2.28	1.50	2.61	1.35

can add the individual losses (this corresponds to the worst case where classification of different bits is independent). From this formula, we can see that 16.8 bits of entropy has been lost from the 48 bits of the DES key used in the first round (out of a maximum possible loss of 32 bits if the classification was perfect). The loss of entropy of the keyspace can be translated into reduced expected computational cost of a guided exhaustive search through the entire keyspace that examines more likely keys earlier than the less likely keys.

For DES implementations, the attack can be improved substantially. Templates can be built not just for round 1, *s*-box output bits but also for other bits such as the data bits fed to the second round. These templates will further narrow down the possibilities for the 48 key bits used in the first round. In addition, templates can be built for the corresponding DPA attacks on the last two rounds of DES (which utilize another 48-bit size subset of the key) and so on. Depending on the implementation, single-bit templates can also be built directly for the key bits that are likely to be highly effective since the same key bits show up in multiple locations in a round and across multiple rounds.

To summarize, single-bit template attacks are capable of classifying a single bit from a single side-channel sample with high probability even though the influence of a single bit on the side-channel signal at a point in time is very small and could be masked by several sources of noise including variation in adjacent bits. Cryptographic algorithms with high contamination properties such as DES are ideally suited for single-bit classification. Multiple precomputed single-bit templates can lead to practical guided keyspace search algorithms using only a single sample from the target device. Moreover, single-bit attacks when combined with other attacks can result in much more devastating attacks such as template-enhanced DPA [3] that use a DPA-like attack technique to overcome the random masking countermeasure, provided an adversary can acquire a single test card with a faulty RNG.

14.5 Improved DPA/DEMA Metric

In Section 14.4, when discussing template attacks, we assumed that the adversary had access to a test device identical to the target device and that he could carry out a profiling stage using the test device. In many circumstances, access to a test device may not be possible. In such cases, a DPA-style attack is preferred since it assumes no prior knowledge of device characteristics or implementation. In this section, we apply theory from Section 14.3 to optimize the analysis of existing single-channel DPA attacks.

14.5.1 Improving DPA

In the traditional DPA attack, an adversary collects a set of N signals, $\mathbf{O}_i, i = 1, \ldots, N$ emanating from a given channel. Assume that the signals are normalized to have zero sample average over all N signals. For each hypothesis H under consideration, the N signals are divided into two bins, termed the 0-bin and the 1-bin, with $N_{H,0}$ and $N_{H,1}$ samples, respectively. Let $\mu_{H,0}[j]$ and $\mu_{H,1}[j]$ be the sample means of signals in the 0-bin and the 1-bin, respectively, for the hypothesis H. The next step in the DPA attack consists of computing the differences of sample means $\mu_H[j] = \mu_{H,0}[j] - \mu_{H,1}[j]$ for all hypotheses and deciding in favor of the hypothesis H_i if $|\mu_{H_i}[j]|$ has the largest peak among all differences of means. In other words, the decision metric for the hypothesis H at time j is given by

$$M_H[j] = \Big(\mu_{H,0}[j] - \mu_{H,1}[j]\Big)^2 \tag{14.5}$$

and the decision is made in favor of the hypothesis H_i if for some value of j, say j_0, $M_{H_i}[j_0] >= M_H[j]$ for all H and j.

The traditional DPA attack and its variations have been successfully applied to attack several cryptographic implementations. However, by using the theory developed in Section 14.3, the effectiveness of traditional DPA can be increased significantly.

Before proceeding further, assume a void hypothesis H_v which corresponds to a random bifurcation of the N signals into the 0-bin and the 1-bin. Using the Gaussian assumption and Equation (14.3), the metric of a hypothesis H_i with respect to the null hypothesis at time j is given by

$$M_{H_i}[j] = \frac{\Big(\mu_{H_i}[j] - E[\mu_{H_v}[j]]\Big)^2}{V[\mu_{H_v}[j]]} - \frac{\Big(\mu_{H_i}[j] - E[\mu_{H_i}[j]]\Big)^2}{V[\mu_{H_i}[j]]} - \ln\Big(\frac{V[\mu_{H_i}[j]]}{V[\mu_{H_v}[j]]}\Big) \tag{14.6}$$

In order to compute this metric, we need the values of the following parameters: $E[\mu_{H_v}[j]]$, $V[\mu_{H_v}[j]]$, $E[\mu_H[j]]$, and $V[\mu_H[j]]$. Since in the DPA attack, the adversary skips the profiling phase of the attack, Equation (14.6) is not directly applicable. In such cases, the theory suggests that unknown parameters of the test equation be estimated directly from the collected signals. If the adversary uses a maximum likelihood estimate of these parameters, then the resulting test is referred to as the generalized maximum likelihood testing.

For the DPA attack, calculating the maximum likelihood estimate of the test parameters involves solving a set of nonlinear coupled equations. Therefore, instead of using the maximum likelihood estimates of these parameters, we use sample estimates as follows: Let $\sigma^2_{H,0}[j]$ and $\sigma^2_{H,1}[j]$ be the sample variances of the signals in the 0-bin and the 1-bin, respectively, at time j for hypothesis H. We propose the following sample estimators[2] of parameters in Equation (14.6):

$$\begin{aligned} E[\mu_H[j]] &= \mu_H[j] \\ V[\mu_H[j]] &= \frac{\sigma^2_{H,0}[j]}{N_0} + \frac{\sigma^2_{H,1}[j]}{N_1} \end{aligned} \tag{14.7}$$

[2] We omit the derivation of these estimators as the derivation is tedious and follows from straightforward algebraic manipulations.

Substituting these in Equation (14.6), we get the following formula for the metric:

$$M_{H_i}[j] = \frac{\left(\mu_{H_i}[j] - \mu_{H_v}[j]\right)^2}{\frac{\sigma^2_{H_v,0}[j]}{N_0} + \frac{\sigma^2_{H_v,1}[j]}{N_1}} - \ln\left(\frac{\frac{\sigma^2_{H_i,0}[j]}{N_0} + \frac{\sigma^2_{H_i,1}[j]}{N_1}}{\frac{\sigma^2_{H_v,0}[j]}{N_0} + \frac{\sigma^2_{H_v,1}[j]}{N_1}}\right) \tag{14.8}$$

Table 14.6 shows the results of applying this method to attacking the s-box lookup for a DES implementation. The first column shows the bit being predicted, the second shows the number of samples required for the correct key hypothesis to emerge as the winner under the traditional DPA metric, while the third column shows the number of samples needed with the new metric. Clearly by using a better metric, our improvement in the DPA attack reduces the number of signals needed by a factor of 1.4–3.

Table 14.6 DPA results, mean difference vs. approximate generalized maximum likelihood.

S-box hyp.	Min samples (mean diff.)	Min samples(Max. Likl.)
S1,B3	640	350
S2,B3	630	210
S7,B3	110	40
S8,B3	130	90

14.6 Multi-Channel Attacks

As we have seen, there are several side-channels including power and multiple EM channels that carry somewhat different information. Given this multitude of information-bearing signals, a natural question to ask is how these multiple leakages could be combined to enable better attacks. In addition, since each additional sensor and side-channel signal used for analysis raises the cost and complexity of an attack, another important question is how a resource-limited adversary could best select the sensors and side-channels to mount an attack. In addition, system designers would like to know how much information could leak to an adversary who is able to place a set of side-channel sensors to capture information from the device. In this section we will use the theory developed in Section 14.3 to answer all these questions.

14.6.1 Multiple Channel Selection

Consider a resource-limited adversary who can select at most M channels for an attack. When viewed in terms of our model, this problem conceptually has a very simple solution: The adversary should choose those M channels that minimize his probability of error in the maximum likelihood testing.

This apparently simple technique can be quite subtle and tricky in practice. Clearly, in situations where a well-prepared adversary has nicely characterized and approximated signals from each of the channels under each hypothesis and the corresponding joint noise probability distribution between all the channels, the adversary can also calculate the error probability for each possible choice of M channels, at least for small M. For example, if the noise is Gaussian and independent of the hypothesis, then from Equation (14.4), since $\mathrm{erfc}(\cdot)$ decreases exponentially with Δ, the goal of an adversary limited to just two channels would be to choose channels in such a manner as to maximize the output signal-to-noise ratio Δ^2.

If instead of a rigorous approach, channels are selected by heuristic techniques, then the resulting selection could be sub-optimal for various subtle reasons. First, different side-channels could leak different aspects of information relative to the hypotheses being tested and sometimes there could be value in combining channels which provide widely dissimilar information rather than combining those which provide similar but partial information. Second, even if many channels provide the same information, picking multiple channels from this set could still be valuable since that may be almost as good as having the ability to make multiple invocations of the device with the same data and collecting a single side-channel. Even for the case where only two side-channels can be selected, the optimal choice is quite tricky and subtle as shown by the example below where the naive approach of choosing the two signals with best signal-to-noise ratios is shown to be sub-optimal.

Example 14.1. Consider the case where an adversary can collect two signals $[O_1, O_2]^T$ at a single point in time, such that under the hypothesis H_0, $O_k = N_k$ for $k = 1, 2$ and under the hypothesis H_1, $O_k = S_k + N_k$. Assume that $\mathbf{N_i} = (N_1, N_2)^T$ has zero mean multivariate Gaussian distribution with

$$\Sigma_N = \begin{pmatrix} 1 \rho \\ \rho\, 1 \end{pmatrix}$$

Note that O_1 and O_2 have signal-to-noise ratios of S_1^2 and S_2^2, respectively. After some algebraic manipulations, we get

$$\Delta^2 = \frac{(S_1+S_2)^2}{2(1+\rho)} + \frac{(S_1-S_2)^2}{2(1-\rho)} \tag{14.9}$$

Now, consider the case of an adversary who discovers two AM-modulated carrier frequencies which are close and carry compromising information, both of which have very high and equally good signal-to-noise ratios ($S_1 = S_2$) and another AM-modulated carrier in a very different band with a lower signal-to-noise ratio. An intuitive approach would be to pick the two carriers with high signal-to-noise ratios. In this case $S_1 = S_2$ and we get $\Delta^2 = 2S_1^2/(1+\rho)$. Since both signals originate from carriers of similar frequencies, the noise that they carry will have a high correlation coefficient ρ, which reduces Δ^2 at the *output*. On the other hand, if the adversary collects one signal from a good carrier and the other from the worse quality carrier in the different band, then the noise correlation is likely to be lower or even 0. In this case

$$\Delta^2 = \frac{(S_1+S_2)^2}{2} + \frac{(S_1-S_2)^2}{2} = S_1^2(1+S_2^2/S_1^2) \qquad (14.10)$$

It is clear that the combination of high and low signal-to-noise ratios would be *a better strategy* as long as $S_2^2/S_1^2 > (1-\rho)/(1+\rho)$. For example, if $\rho > 1/3$, then choosing carriers from different frequency bands with even half the signal-to-noise ratio results in better hypothesis testing. □

14.6.2 Multi-Channel Template Attacks

Just as the template attack is the optimal attack strategy for the single-channel case, a multi-channel template attack is the optimal strategy for the multi-channel case. Expanding the template approach to multiple channels is straightforward. For multiple channels, the template attack is identical except that the signals from the multiple channels are concatenated together to yield a larger signal, i.e., for each invocation, a combined signal is created by concatenating the signals from the individual observed channels. Notice that the process of identifying the time instances and sample points could end up selecting somewhat different time slices for each channel, depending purely on the nature of leakage in each channel. The maximum likelihood testing will pick up information from all channels (possibly at different times) for classification.

To show that multiple channels help the classification process, we invoke an operation on the smart card S with two different input bytes and look at just three cycles during which the input was first processed. We collected EM and power samples simultaneously and evaluated how well the template attack could classify a single EM/power trace into the two hypotheses H0 and H1 for the input byte. We did this classification first using exactly one of the power/EM channels and then performed the classification using both channels simultaneously. Figure 14.7 shows the mean EM and power signals for these hypotheses during

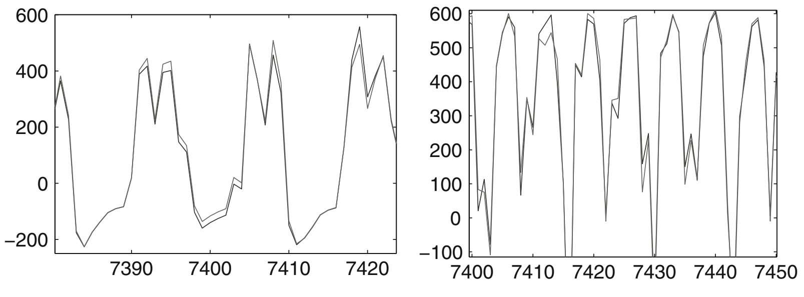

Fig. 14.7 Mean power and EM signals during three cycles for two hypothesis.

Table 14.7 Signal classification error using power, EM, and combination of power and EM.

Correct hypothesis	Error (Pwr)	Error (EM)	Error (EM+Pwr)
H0	9.5%	15.1%	2.8%
H1	20.1%	15.2%	6.6%

these three cycles side by side.[3] Table 14.7 shows the error rate of our classification effort for inputs belonging to each hypothesis. One can clearly notice that using both channels simultaneously results in better classification compared to any single channel.

14.6.3 Multi-Channel DPA

Multi-channel DPA attack is a generalization of the single-channel DPA attack. In this case, the adversary collects N signals, $\mathbf{O}_i, i = 1,\ldots,N$. In turn, each of the signals $\mathbf{O}_i$ is a collection of L signals collected from L side-channels. Thus, $\mathbf{O}_i = [\mathbf{O}_i^1,\ldots,\mathbf{O}_i^L]^T$, where $\mathbf{O}_i^l$ represents the ith signal from the lth channel. Note that all DPA-style attacks treat each time instant independently and leakages from multiple channels can only be pooled together if they occur at the same time. Thus, in order for multi-channel DPA attacks to be effective, the selected channels must have very similar leakage characteristics.

The formulae for computing the metric for multi-channel DPA attack are generalizations of those for the single channel as described in Section 14.5. The main difference is that the expected value of sample mean difference at time j under hypothesis H is a vector of length L, with the lth entry being the sample mean difference of the lth channel. Furthermore, the variance of the b-bin under hypothesis H at time j is a covariance matrix of size $L \times L$ with the i, jth entry being the correlation between signals from the ith and jth channels. Once again, as in the DPA attack, the adversary does not have the luxury of estimating these parameters. Therefore, we substitute sample estimates for these parameters along the same lines as in Equation (14.7). We skip the cumbersome formulae and directly go to the results of multi-channel DPA attacks.

Table 14.8 shows sample results of an attack on the s-box lookups in a DES implementation using the power channel together with an EM channel whose leakage is similar to the power channel. The first column shows the bit being predicted, the second shows the number of signals required for the correct key hypothesis to emerge as the winner using both channels with the multi-channel metric, the last two columns show the number of signals needed for the power and EM channels separately using the new DPA/DEMA metric. From this it is clear that the number of invocations needed for two-channel attacks can be significantly less compared to single-channel attacks.

[3] The slight offset in time is due to delay of EM signals with respect to the power signal.

Table 14.8 Multi-channel DPA-style attack using power, EM, and power and EM.

S-box hyp.	Min samples (Pwr+EM)	Min samples (Pwr)	Min samples (EM)
S1,B1	150	170	640
S1,B2	60	(>1000)	340
S1,B3	110	350	160
S2,B2	30	50	230
S2,B3	120	210	340
S4,B0	60	200	340
S6,B1	180	180	190
S7,B3	30	40	520
S8,B3	60	90	140

14.7 Toward Information Leakage Assessment

In this section, we address the following question: Can the information obtained by combining leakages from several (or even all possible) signals from available sensors be quantified regardless of the signal processing capabilities and computing power of an adversary?

We will use maximum likelihood testing to craft a methodology to assess information leakage from elementary operations in a device. This methodology takes into account the power signals and all EM signals extractable from all the given sensors across the entire EM spectrum. Results of such an assessment will enable one to bound the success probability of the optimal adversary for any given hypothesis.

Assume that for a single invocation, the adversary captures the power signal and emanations across the entire electromagnetic spectrum from all sensors in an observation vector $\mathbf{O}$. Let Ω denote the space of all possible observation vectors $\mathbf{O}$. Since the likelihood ratio, $\Lambda(\mathbf{O})$, is a function of the random vector $\mathbf{O}$, the best achievable success probability, P_s, is given by

$$P_s = \sum_{\mathbf{O}\in\Omega} I_{\{\Lambda(\mathbf{O})>1\}} p_{\mathbf{N1}}(\mathbf{O}-\mathbf{S}_1) + I_{\{\Lambda(\mathbf{O})<1\}} p_{\mathbf{N0}}(\mathbf{O}-\mathbf{S}_0) \tag{14.11}$$

where I_A denotes the indicator function of the set A and $p_{\mathbf{N1}}(\mathbf{O}-\mathbf{S}_1)$ and $p_{\mathbf{N0}}(\mathbf{O}-\mathbf{S}_0)$ are noise distributions under the hypothesis 1 and 0, respectively.

When the adversary has access to multiple invocations, an easier way of estimating the probability of success/error involves a technique based on moment generating functions. We begin by defining the logarithm of the moment generating function of the likelihood ratio:

$$\mu(s) = \ln\Big(\sum_{\mathbf{O}\in\Omega} p^{s}_{\mathbf{N1}}(\mathbf{O}-\mathbf{S}_1)\, p^{1-s}_{\mathbf{N0}}(\mathbf{O}-\mathbf{S}_0)\Big) \tag{14.12}$$

The following is a well-known result from information theory:

Fact 2 *Assume we have several statistically independent observation vectors*[4]

$$\mathbf{O}_1, \mathbf{O}_2, \ldots, \mathbf{O}_L$$

For this case, the best possible exponent in the probability of error is given by the Chernoff information*:*

$$C \stackrel{\text{def}}{=} -\min_{0 \le s \le 1} \mu(s) \stackrel{\text{def}}{=} -\mu(s_m) \tag{14.13}$$

Note that $\mu(\cdot)$ is a smooth, infinitely differentiable, convex function and therefore it is possible to approximate s_m by interpolating in the domain of interest and finding the minima. Furthermore, under certain mild conditions on the parameters, the error probability can be approximated by

$$P_\varepsilon \approx \frac{1}{\sqrt{8\pi L \mu''(s_m) s_m (1-s_m)}} \exp(L\mu(s_m)) \tag{14.14}$$

Note that in order to evaluate Equation (14.11) or (14.14), we need to estimate $p_{\mathbf{N0}}(\cdot)$ and $p_{\mathbf{N1}}(\cdot)$. In general, this can be a difficult task. However, by exploiting certain characteristics of the CMOS devices, estimation of $p_{\mathbf{N0}}(\cdot)$ and $p_{\mathbf{N1}}(\cdot)$ can be made more tractable.

14.7.1 Practical Considerations

We will now outline some of the practical issues associated with estimating $p_{\mathbf{N0}}(\cdot)$ and $p_{\mathbf{N1}}(\cdot)$ for any hypothesis. The key here is to estimate the noise distribution for each cycle of each elementary operation and for each relevant state R that the operation can be invoked with. This results in the signal characterization, $\mathbf{S}_R$, and the noise distribution, $p_{\mathbf{NR}}(\cdot)$, which is sufficient for evaluating $p_{\mathbf{N0}}(\cdot)$ and $p_{\mathbf{N1}}(\cdot)$.

There are two crucial assumptions that facilitate estimating $p_{\mathbf{NR}}(\cdot)$: first, on chip-cards examined by us the typical clock cycle is 270 ns. For such devices, most of the compromising emanations are well below 1 GHz which can be captured by sampling the signals at a Nyquist rate of 2 GHz. This sampling rate results in a vector of 540 points per cycle per sensor. Alternatively, one can also capture all compromising emanations by sampling judiciously chosen and slightly overlapping bands of the EM spectrum. The choice of selected bands is dictated by considerations such as signal strength and limitations of the available equipment. Note that the slight overlapping of EM bands would result in a corresponding increase in the number of samples per clock cycle; however, it remains in the range of 600–800 samples per sensor.

[4] For simplicity, this chapter deals with *independent* elementary operation invocations. Techniques also exist for adaptive invocations.

The second assumption, borne out in practice (see [6]), is that for a fixed relevant state, the noise distribution $p_{\mathbf{NR}}(\cdot)$ can be approximated by a Gaussian distribution. This fact greatly simplifies the estimation of $p_{\mathbf{NR}}(\cdot)$ as only about one thousand samples are needed to roughly characterize $p_{\mathbf{NR}}(\cdot)$. Moreover, the noise density can be stored compactly in terms of the parameters of the Gaussian distribution.

These two assumptions imply that in order to estimate $p_{\mathbf{NR}}(\cdot)$ for a fixed relevant state R, we need to repeatedly invoke (say 1000 times) an operation on the device starting in the state R and collect samples of the emanations as described above. Subsequently, the signal characterization S_R can be obtained by averaging the collected samples. The noise characterization is obtained by first subtracting S_R from each of the samples and then using the Gaussian assumption to estimate the parameters of the noise distribution.

The assessment can now be used to bound the success of any hypothesis testing attack in our adversarial model. For any two given distributions B_0 and B_1 on the relevant states, the corresponding signal and noise characterizations

$$S_0, S_1, p_{\mathbf{N0}}(\cdot), \text{ and } p_{\mathbf{N1}}(\cdot)$$

are a *weighted sum* of the signal and noise assessments of the constituent relevant states S_R and $p_{\mathbf{NR}}(\cdot)$. The error probability of maximum likelihood testing for a single invocation or its exponent for L invocations can then be bounded using Equations (14.11) and (14.13), respectively.

We now give a rough estimate of the effort required to obtain the leakage assessment of an elementary operation. The biggest constraint in this process is the time required to collect samples from approximately 1000 invocations for each relevant state of the elementary operation. For an r-bit machine, the relevant states of interest are approximately 2^{2r}; thus, the leakage assessment requires time to perform approximately $1000 * 2^{2r}$ invocations. Assuming that the noise is Gaussian and that each sensor produces an observation vector of length 800, for n sensors the covariance matrix Σ_N has $(800 * n)^2$ entries. It follows that the computation burden of estimating the noise distribution would be proportional to $(800 * n)^2$. Such an approach is certainly feasible for an evaluation agency from both a physical and computational viewpoint, as long as the size of the relevant state, r, is small.

14.8 Projects

Pre-requisite: Exercises 1–4: A DPA setup for smart cards and sample smart cards. Exercises 1, 2, and 3 require the following data collection:

Capture a set S_A of a few thousand power signals from a smart card with DPA-countermeasures turned off, operating on some fixed input A. Capture another set of signals S_B (where $|S_B| = |S_A|$) from the same smart card operating with another fixed input B, which is very similar to A (e.g., A and B could differ only on one bit). Partition the sets S_A and S_B into two equal-sized training sets S_{A1} and S_{B1} and two equal-sized testing sets S_{A2} and S_{B2}, such that S_{A2} and S_{B2} contain at least a few hundred signals each.

1. *Single Point Binary Classification:* Compute the mean signals μ_A and μ_B using the sets S_{A1} and S_{B1}. Subtract μ_A from μ_B and plot the difference of means signal to locate the first point in time P where there is a significant difference (or peak). At this point P, signals in S_{A1} have a mean μ_{AP} and signals in S_{B1} have a mean μ_{BP}. Use the set S_{A1} to also compute the variance σ^2_{AP} of these signals at point P and likewise compute σ^2_{BP} from S_{B1}. Make the assumption that the signals from S_{A1} and S_{B1} are normally distributed at point P (Gaussian assumption, with $n = 1$). Next use the maximum likelihood principle and Gaussian assumption about point P to classify signals from testing sets S_{A2} and S_{B2} by just looking at point P. Compute the fraction of correct/incorrect classification from sets S_{A2} and S_{B2}.
2. *Multi-Point Binary Classification, Univariate Statistics:* Build upon experiment 1 above by considering two other peaks that are located near the first peak P. In this exercise, compute the means and variances at these three points for inputs A and B by using the training sets S_{A1} and S_{B1}. Then use the maximum likelihood principle with univariate Gaussian statistics (i.e., assuming that the noise co-variance matrix across these three points is diagonal) to classify the test signals from S_{A2} and S_{B2} using these means and variances only. Compute the fraction of correct/incorrect classification from sets S_{A2} and S_{B2}.
3. *Multi-Point Binary Classification, Multivariate Statistics:* Repeat experiment 2 above with multivariate statistics, i.e., by computing the covariance between the noise at the three different points in time and using the maximum likelihood principle and Gaussian assumption to classify the test signals. Compute the fraction of correct/incorrect classification from sets S_{A2} and S_{B2}. Compare the results of experiments 1, 2, and 3 to see how adding additional information (points) and better analysis (multivariate statistics) improves the classification accuracy.
4. *Comparison of Metrics for DPA:* Run a DPA experiment on your smartcard with the usual DPA metric (Equation 14.5). Run the same experiment with the metric provided in Equation (14.8). Compare the results in terms of number of signals needed to obtain the correct result of the DPA analysis.
5. *Signal Selection:* Suppose you can collect two signals $[O_1, O_2]$ at a point in time (see Example 14.1) and use these two signals to determine between two hypotheses: H_0, $O_k = N_k$, for $k = 1,2$, and under the hypothesis H_1, $O_k = S_k + N_k$. Assume that $\mathbf{N_i} = (N_1, N_2)^T$ has zero mean multivariate Gaussian distribution with

$$\Sigma_N = \begin{pmatrix} 1 \rho \\ \rho 1 \end{pmatrix}$$

Signal O_1 has already been selected and $O_1 = S_1 + N_1$. You have three choices for the second signal O_2:

1. Choice 1: $S2 = 0.9 * S1, \rho = 0.5$.
2. Choice 2: $S2 = 0.8 * S1, \rho = 0.2$.
3. Choice 3: $S2 = 0.65 * S1, \rho = 0$.

Which of these signals should you choose for O_2 and why?

References

1. O. Acıicmez, Ç. K. Koç, and J.-P. Seifert. Predicting secret keys via branch prediction. In M. Abe editor, *Topics in Cryptology CT-RSA 2007, The Cryptographers Track at the RSA Conference 2007*, pp. 225–242, Springer-Verlag, Lecture Notes in Computer Science series 4377, 2007.
2. D. Agrawal, B. Archambeault, J. R. Rao, and P. Rohatgi. The EM side-channel(s). In B. Kaliski, Ç. K. Koç, and C. Paar editors, *Proceedings of CHES 2002*. Lecture Notes in Computer Science, vol. 2523, pp. 29–45, Springer, 2002.
3. D. Agrawal, J. R. Rao, P. Rohatgi, and K. Schramm. Templates as Master Keys. In J. R. Rao and B. Sunar editors, *Proceedings of CHES 2005*, Lecture Notes in Computer Science, vol. 3659, pp. 15–29, Springer, 2005.
4. D. Asinov and R. Agrawal. Keyboard acoustic emanations. In *Proceeding of the IEEE Symposium on Security and Privacy 2004*, pp. 3–11, 2004.
5. D. J. Bernstein. Cache-timing attacks on AES. Technical Report, p. 37, April 2005, available at http://cr.yp.to/antiforgery/cachetiming-20050414.pdf
6. S. Chari, J. R. Rao, and P. Rohatgi. Template attacks. In B. Kaliski, Ç. K. Koç, and C. Paar editors, *Proceedings of CHES 2002*, Lecture Notes in Computer Science, vol. 2523, pp. 13–28 Springer, 2002.
7. K. Gandolfi, C. Mourtel, and F. Olivier. Electromagnetic analysis: Concrete results. In Ç. K. Koç, D. Naccache, and C. Paar editors, *Proceedings of CHES 2001*, Lecture Notes in Computer Science, vol. 2162, pp. 251–261, Springer, 2001.
8. P. C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In N. Koblitz editor, *Advances in Cryptology – CRYPTO '96*, Lecture Notes in Computer Science, vol. 1109, pp. 104–113, Springer-Verlag, 1996.
9. P. C. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In M. Wiener editor, *Proceedings of Advances in Cryptology CRYPTO '99*, Lecture Notes in Computer Science, vol. 1666, pp. 388–397, Springer-Verlag, 1999.
10. M. Kuhn. Optical Time-domain eavesdropping risks of CRT displays. In *Proceedings of the Symposium on Security and Privacy*, pp. 3–18, 2002.
11. J. Loughry and D. Umphress. Information leakage from optical emanations. In *ACM Transactions on Information and System Security*, vol. 5, pp. 262–289, 2002.

12. D. A. Osvik, A. Shamir, and E. Tromer. Cache attacks and countermeasures: The case of AES. In D. Pointcheval editor, *Topics in Cryptology CT-RSA 2006, The Cryptographers Track at the RSA Conference 2006*, pp. 1–20, Lecture Notes in Computer Science, vol. 3860, Springer-Verlag, 2006.
13. C. Percival. Cache missing for fun and profit. In *BSDCan 2005*, Ottawa, 2005, available at http://www.daemonology.net/hyperthreading-considered-harmful/
14. J.-J. Quisquater and D. Samyde. Electromagnetic analysis (EMA): Measures and countermeasures for smart cards. In *Proceedings of e-Smart 2001*, Lectures Notes in Computer Science (LNCS), vol. 2140, pp. 200–210, Springer, 2001.
15. H. L. Van Trees. *Detection, Estimation, and Modulation Theory*, Part I. John Wiley & Sons, New York, 1968.