Towards Sound Approaches to Counteract Power-Analysis Attacks

Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi

I.B.M. Thomas J. Watson Research Center,
Yorktown Heights, NY 10598, U.S.A. Email:
{schari,csjutla,jrrao,roha}@watson.ibm.com

Abstract. Side channel cryptanalysis techniques, such as the analysis of instantaneous power consumption, have been extremely effective in attacking implementations on simple hardware platforms. There are several proposed solutions to resist these attacks, most of which are ad–hoc and can easily be rendered ineffective. A scientific approach is to create a model for the physical characteristics of the device, and then design implementations provably secure in that model, i.e, they resist generic attacks with an *a priori* bound on the number of experiments. We propose an abstract model which approximates power consumption in most devices and in particular small single–chip devices. Using this, we propose a generic technique to create provably resistant implementations for devices where the power model has reasonable properties, and a source of randomness exists. We prove a lower bound on the number of experiments required to mount statistical attacks on devices whose physical characteristics satisfy reasonable properties.

1 Introduction

Side channel cryptanalysis *i.e.*, cryptanalysis using information leaked during the computation of cryptographic primitives has successfully been used to break implementations on simple platforms such as chip–cards[1, 2, 8]. It has been claimed [2] that in chip–card like devices, *all* straightforward implementations are susceptible to attack by power analysis techniques. Analogous to timing attacks[1], simple power attacks, where the adversary extracts key bits by identifying the execution sequence from the instantaneous power consumption, are easier to protect against by making the execution sequence independent of the key bits. Differential and Higher order Differential power attacks, on the other hand are extremely powerful and difficult to protect against. These attacks rely on the ability of the attacker to create two different statistical distributions on the values being manipulated during a single instruction (or a set of instructions) based on known input/output and guesses of few key bits. If these distributions can be distinguished, using statistical tests on instantaneous power samples (or any other side channel), then the attacker can verify the key guesses.

Due to the wide–ranging impact of these attacks, there have been several proposed commercial implementations which claim to resist these and similar attacks. Without rigorous justification, several of these solutions are ad–hoc

Michael Wiener (Ed.): CRYPTO'99, LNCS 1666, pp. 398-412, 1999.

and based on simplistic techniques such as probabilistically reordering execution paths. It is our view that such approaches essentially miss the import of these attacks and their underlying basis. Furthermore, these simple countermeasures can be nullified by signal processing. Instead, we propose that the focus should be on sound scientific approaches to this problem *i.e.* develop an accurate and abstract model of the problem and identify rigorously proven techniques which can be used as effective countermeasures. A realistic goal for a sound and effective countermeasure is to *provably* resist all generic attacks by an adversary who is allowed to perform and observe at most an *a priori* fixed number of adaptively chosen operations. Generic attacks are those that use functional specifications and generic physical characteristics and do not depend on specific implementations and devices. Typically, simple single chip devices and the keys within them are short–lived and the number of key–dependent operations performed by them over their lifetime is also limited. A countermeasure secure against an *a priori* bound on the number of experiments, can be used in practice by explicitly enforcing this upper bound in the devices themselves.

In this paper, we propose a general, simplified model for the power consumption in simple devices, and use this to restate the basis of these attacks and to analyze countermeasures. We examine ad–hoc approaches which have been proposed and discuss why they are easy to defeat. A general technique is then proposed as a countermeasure against statistical attacks in devices where the power model is reasonable and a source of randomness is available. This technique is based on well known secret sharing schemes where each bit of the original computation is divided probabilistically into shares such that any proper subset of shares is statistically independent of the bit being encoded and thus, yields *no* information about the bit. Computation of cryptographic primitives is done accessing only the random shares at each point, with intermediate steps computing only the shares of the result. Splitting the bit into multiple probabilistic shares amplifies the uncertainty of the adversary at each point and forces him to work with the joint distributions of the signal at the points where the shares are being accessed. For computation of common cryptographic primitives, simple sharing schemes based on XOR and addition modulo 2^8 can be used.

We make realistic assumptions about the power consumption model for devices with respect to the uncertainty of the adversary at each point, and analyze the efficacy of this technique to withstand power analysis attacks. Using this, we rigorously prove lower bounds on the number of observations required to statistically distinguish distributions, defined in power attacks, using observations on power samples. Our lower bounds are exponential in the number of shares that each bit (or byte) of the computation is encoded by. The models and lower bounds are initial steps in developing a formal framework for the problem of computation in the presence of the information leaked due to the observations on the physical characteristics of devices. Only solutions which can be proved secure in a formalized model should be considered for implementation. Substantial effort is still required to find more appropriate models and stronger analysis.

Section 2 describes a formalization of a general power consumption model.

Section 2.1 restates power analysis attacks in this framework. In section 3, we analyze countermeasures, examine simple ad-hoc solutions which are ineffective, and propose the secret sharing scheme as a general countermeasure against these attacks. In Section 3.4, we make realistic approximations on the model and rigorously prove lower bounds on the number of samples needed to mount differential power attacks against implementations with this countermeasure.

2 Power Model and Definitions

CMOS devices consume power only when changes occur in logic states, while no significant power is needed to maintain a state. Examples of changes include changes in the contents of the RAM, internal registers, bus-lines, states of gates and transistors etc. In simple chips almost all activity is triggered by an internal/external clock edge and all activity ceases well before the next clock edge. A few processes, such as on-chip noise generators, operate independently of the clock and consume a small, possibly random amount of power continuously. Each clock edge triggers a sequence of power consuming events within the chip, as dictated by the microcode, bringing it to the next state. This sequence depends on parts of the current state of the processor and parts of the state of other subsystems accessed in that cycle. We define *relevant* state bits as the bits of the overall state which determine the sequence of events, and hence the power, during a clock cycle. Depending on the cycle, the relevant state bits could include bits of internal registers, bits on internal and external buses, address bits and contents of memory locations being accessed etc.

The instantaneous power consumption of the chip shortly after a clock edge is a combination of the consumption components from each of the events that have occurred since the clock edge. Each event's timing and power consumption depends on physical and environmental factors such as the electrical properties of the chip substrate, layout, temperature, voltage etc., as well as coupling effects between events of close proximity. As a first approximation, we ignore coupling effects and create a linear model, i.e., we assume that the power consumption function of the chip is simply the sum of the power consumption functions of all the events that take place.

Consider a particular cycle of a particular instruction in the execution path of some fixed code. At the start of the cycle, the chip is in one of several relevant states (determined by the value of the relevant state bits) depending on the input and processing done in earlier cycles. Let $\mathcal{S}$ denote the set of possible relevant states when control reaches this cycle and let $\mathcal{E}$ be the set of all possible events that can occur in a cycle. For each $s \in \mathcal{S}$, and each $e \in \mathcal{E}$, let $occurs(e, s)$ be the binary function which is 1 if e occurs when the relevant state is s and 0 otherwise. Let $delay(e, s)$ be the time delay of the occurrence of event e in state s from the clock edge and let $f(e, t)$ denote the power consumption impulse function of event e with respect to time t ($t = 0$ when e occurs and $f(e, t) = 0$ for $t < 0$). In our linear model, $P(s, t)$, the power consumption function of the chip in that

cycle with state s and time t after the clock edge can be written as

$$P(s,t) = \sum_{e \in \mathcal{E}} f(e, t - delay(e,s)) * occurs(e,s)$$

Due to the presence of noise and asynchronous power consuming components, a better model is:

$$P(s,t) = \mathcal{N}_c(t) + \sum_{e \in \mathcal{E}} (f(e, t - delay(e,s) + \mathcal{N}_d(e,s)) + \mathcal{N}(e,t)) * occurs(e,s) \quad (1)$$

where $\mathcal{N}(e,t)$ is a Gaussian noise component associated with the power consumption function of e, $\mathcal{N}_d(e,s))$ is a Gaussian noise component affecting the delay function and $\mathcal{N}_c$ is the Gaussian external noise component.

2.1 Statistical Power Attacks

Equation 1 shows the strong dependence between the power consumption function and the relevant state which is the basis of all statistical power attacks. Let P_1 and P_2 be two different probability distributions on the relevant state before the clock edge of a certain cycle. From equation 1, it is very likely that the distribution of the instantaneous power when the state is drawn from P_1 will be different from the distribution of the instantaneous power when the state is drawn from P_2. This difference and the distinguishability of different distributions by statistical tests on power samples, is the basis for Differential Power attacks (DPA). Simple distributions are sufficient to mount these attacks. For example, in the DPA attacks described in [2], P_1 and P_2 are very simple: P_1 is the uniform distribution on the set of all relevant states which have a particular relevant state bit 1 and P_2 is the uniform distribution on the set of all relevant states which have the same bit 0. The difference in the power distribution for these two cases represents the effect of that particular relevant state bit on the net power consumption. This can be used to extract cryptographic keys by guessing parts of keys, using this to predict a relevant state bit and defining the distributions as described above. Higher order differential attacks are those in which the distributions P_1 and P_2 are defined over multiple internal state variables and where the adversary has access to multiple side channels. Appendix 1 shows an example of distributions induced in the power consumption signal by distributions on the relevant state, for an actual chip–card.

In defining security against statistical attacks we use the strongest possible notion: using the side channel, with high probability, the adversary should not be able to predict with even a slight advantage, any bit that he could not predict from just the knowledge of inputs, outputs and program code. In using the side channel, the adversary is limited to trying to distinguish distributions which he can affect by choice of inputs and selection based on outputs. These are limited to distributions on bits such as bits in the algorithm specification e.g., bits of the key, bits which depend directly on the key and the input etc. Also, in an attack against the implementation the adversary could affect deterministic temporary

variables, registers etc. We informally define the set of realizable distributions which the adversary can directly affect as follows:

Definition 1. A distribution on state bits is *realizable* if the adversary can induce the distribution by suitable choice of inputs and selection based on outputs. In particular, this excludes distributions of state bits which result from explicit randomization introduced in the implementation outside of the specification.

The ability to distinguish any two realizable distributions is potentially advantageous to the adversary. Using standard notations (see for example[7]) we define the distinguishing probability of an adversary as follows:

Definition 2. Let M be a binary valued adversary who adaptively chooses k inputs and has access to the side channel signals for the corresponding operations. Let B_1 and B_2 be any two realizable distributions on the bits of a computation, and D_1 and D_2 the distributions induced on the side channel signals, by the choice of inputs and B_1 and B_2 respectively. Let M^D denote M's output when given k input/output pairs and corresponding side–channel samples from a distribution D. The *distinguishing* probability of M when given samples from distributions D_1 and D_2 is $|\Pr(M^{D_1} = 1) - \Pr(M^{D_2} = 1)|$. M is said to distinguish B_1 from B_2 using k side-channel samples, if the distinguishing probability of M, on D_1 and D_2, is at least some constant c.

Using this definition of adversaries, we define a secure computation. We intend to capture *extra* information that the adversary obtains from the side channel.

Definition 3. A computation is said to be secure against N sample side channel cryptanalysis, if for all adversaries M and all realizable distributions B_1 and B_2, if M can distinguish B_1 from B_2 using fewer than N samples, then M can distinguish B_1 and B_2 without the side channel.

The attacks described by [2] can be restated as using the side channel to distinguish distributions B_1 and B_2 which correspond to almost uniform distributions on a few relevant state bits, with a particular state bit (depending on the input and key) being 0 and 1 respectively. There the adversary bases its decision by comparing the mean of the given samples, with some known threshold.

3 Countermeasures to Power Analysis

Using these formal definitions of side channel cryptanalysis, we discuss general countermeasures against such attacks. First, we examine several ad–hoc approaches to fixing this problem, which, we believe, miss the import of these attacks and can easily be rendered ineffective. We present a probabilistic encoding scheme with which we can effectively perform secure computations. Based on realistic approximations of the power models of Section 2 we prove lower bounds on the number of samples required to distinguish distributions.

3.1 Ad–hoc Approaches

Due to the commercial impact, several ad-hoc solutions are currently being implemented and claim to be resistant to these statistical attacks. Unfortunately, most can be defeated by signal processing in conjunction with only moderately more samples. Allowing for about 1 million possible experiments, it is reasonable to assume that the adversary can exploit every relevant state bit in any instruction to mount a statistical attack, provided he can efficiently predict that bit in a significant fraction of the runs based on the code specification, known inputs and small number of guesses for parts of the key.

Some approaches to protecting computation use simple countermeasures such as "balancing", i.e., try to negate the effects of one set of events by another "complementary" set. For example, by ensuring that all bytes used in computation have Hamming weight 4, one can try to negate the effect of each 1 bit by a corresponding 0 bit. Such approaches fail at high resolution and large number of samples, because the power consumption functions and timing of two "complementary" events will be slightly different and the adversary can maximize these differences by adjusting the operating conditions of the card. Another popular approach is to randomize the execution sequence *i.e.* keep operations the same, but permute the order *e.g.* in DES, the S–boxes are looked up in a random order. Unless this random sequencing is done extensively throughout the computation, which may be impossible since the specification forces a causal ordering, it can be undone and a canonical order re–created by signal processing. Attacks can be mounted on the re–ordered signals. Even if the entire computation cannot be canonically reordered, it is sufficient to identify "corresponding" sample points in different runs so that a significant fraction are samples from the same power function P for the same cycle. All statistical attacks that work for P are also applicable to "corresponding" points, although more samples would be needed due to "noise" introduced by unrelated samples. In the case of permuted S-boxes, if the permutation is random, in $\frac{1}{8}$ of the runs S-box 1 is looked up first, and in the remaining samples, the signal at this point, corresponding to different lookups, is essentially random. Thus, even with no reordering, we now have a signal which is attenuated by a factor of 8. Mounting the original attack with 64 times the number of samples yields the same results. Elementary reordering substantially reduces this factor. A similar countermeasure in hardware is typically achieved by making instructions take a variable number of cycles or by having the cycles be of varying length(see [4]). Once again, it is very easy to negate all these countermeasures with signal processing.

3.2 A general countermeasure

A general countermeasure is to ensure that the adversary cannot predict any relevant bit in any cycle, without making run–specific assumptions independent of the actual inputs to the computation. This makes statistical tests involving several experiments impossible, since the chance of the adversary making the correct assumptions for each run is extremely low. While this yields secure

computation, it is not clear how one can do effective computation under this requirement since no bit depending directly on the data and key can be manipulated at any cycle. In some cases the function being computed has algebraic properties that permits such an approach, e.g., for RSA one could use blinding [1, 3] to partially hide the actual values being manipulated. Another class of problems where this is possible is the class of random self–reducible problems [9]. Such structure is unlikely to be present in primitives such as block ciphers.

3.3 Encoding

The encoding we propose is to randomly split every bit of the original computation, into k shares where each share is equiprobably distributed and every proper subset of $k-1$ shares is statistically independent of the encoded bit. Computation can then be carried securely by performing computation only the shares, without ever reconstructing the original bit. Shares are refreshed after every operation involving them to prevent information leakage to the adversary.

To fix a concrete encoding scheme, we assume that each *bit* is split into k shares using any scheme which has the required stochastic properties. For instance, bit b can be encoded as the k shares $b \oplus r_1, r_2, \ldots, r_{k-1}, r_1 \oplus \ldots \oplus r_{k-1}$, where the r_is are randomly chosen bits. Furthermore, assume that each share is placed in a separate word at a particular bit position and all other bits of the share word are chosen uniformly at random.

In practice, it would be more useful, if each word of computation is split similarly into k shares. In that case, other schemes of splitting into shares based on addition mod 2^8, subtraction mod 2^8 would also be viable. Encoding bytes of data manipulated by splitting them into shares would yield the optimal performance. Ignoring the initial setup time, the performance penalty in performing computation using just the k shares is a factor of k. Our results which have been proved based on the bit encoding scheme would also work for this case but the bounds they yields are based only on the characteristics of the noise within the chip, and hence may not be optimal. This is discussed briefly after the analysis for the bit encoding case. The results and analysis we present here can serve as a framework in which to prove results for the byte encoding scheme.

The method to encode the bit in secret shares should be chosen based on the computation being protected. For instance, for an implementation of DES, the XOR scheme is ideal since the basic operations used are XOR, permutations, and table lookups. Table lookups can be handled by first generating a random rearrangement of the original table since a randomized index will be used to look up the table. This step increases the overhead beyond the factor of 2.

In practice, the splitting technique needs to be applied only for a sufficient number of steps into the computation until the adversary has very low probability of predicting bits, i.e., till sufficient secret key dependent operations have been carried out. Similar splitting also has to be done at end of the computation if the adversary can get access to its output. For instance, in DES, one needs to use the splitting scheme only for the first four and last four rounds.

3.4 Analysis

We analyze the encoding scheme described above, by making reasonable assumptions on distribution of side channel information and prove that the amount of side channel information required grows exponentially in k, the number of shares. For concreteness we fix the XOR bit encoding scheme and consider the instantaneous power consumption at some time instant in a cycle manipulating a share. The relevant state in that cycle will not only include a share of the bit, but also all the other random bits in the word. It is quite reasonable to assume that the contributions of all bits in the word will be similar in magnitude. From equation (1), expanding $occurs(e, s)$ as a linear form over the bits of s, the instantaneous power consumption when a particular share is being manipulated will be

$$P = b \times s_0 + P \times s_0 + R$$

where b is the contribution of just the shared bit s_0, $P \times s_0$ is the distribution of power contributions of events which require s_0 and other state bits and finally, R is the distribution of events which are independent of the bit s_0. In operations such as load, store and XOR, if s_0 is a bit in a word being manipulated, the factor P can be viewed as a small perturbation on the real value b. In simple operations there is no "interaction" between the different bits of the value being manipulated and an approximation, we will ignore the contribution of the variable P. The random variable R is typically much larger than b since it includes the sum of similar contributions from all other bits. For most operations, R is the sum of almost independent distributions which is very well approximated by the Normal Distribution. Thus, we make the realistic assumption, which has been empirically tested as shown in Appendix 1, that R has a normal distribution with mean μ and variance σ^2. The results we prove can also be shown to hold in the case that R is the sum of i.i.d's, which is the case for operations such as load, store, XOR. Further work needs to be done to analyze more complex and precise distributions which model all chip–card operations such as multiply where there is interaction between the bits being manipulated and it is unlikely that one can ignore the contribution of the variable P.

Assume that in each sample the adversary has access to the k signal values corresponding to the power consumption at instructions which access the shares $b \oplus r_1, r_2, \ldots, r_{k-1}, r_1 \oplus r_2 \ldots \oplus r_{k-1}$. Rewrite these bits as $r_1, \ldots r_k$, with $r_1 \oplus \ldots \oplus r_k = b$. Denote the distribution of the instantaneous power consumption signal at these points by random variables $Z_1, Z_2, \ldots Z_k$. Also, let $Z_i = A_i + X_i$, where A_i is the contribution due to the bit in concern and X_i is the additive factor which follows the distribution R. By the definition of the encoding, A_i takes values 0 and 1 with probability $\frac{1}{2}$ each. Any noise in the contribution due to A_i can be absorbed in R without affecting the distribution of R since R is typically much bigger than b. Thus, the power contribution due to A_i is 1 if $r_i = 1$ (and 0 if $r_i = 0$). It is important to note that the A_i's are not independent since $A_1 \oplus \ldots \oplus A_k = b$.

In defining distributions that an adversary can try to distinguish using inputs, outputs and the side channel information, note that the adversary can not control

the randomizing variables r_i's. Thus, the only realizable distributions are those with the value of the bit b being 0 and 1 *i.e.* distributions D_1 and D_2 where: a random variable Y sampled from D_1 is given by

$$\langle A_1 + X_1, \ldots, A_k + X_k \rangle$$

with the condition that $A_1 \oplus \ldots \oplus A_k = b$, while a random variable sampled from D_2 is given by

$$\langle A_1 + X_1, \ldots, A_k + X_k \rangle$$

with the condition that $A_1 \oplus \ldots \oplus A_k = 1 \oplus b$. This is more general than the encoding scheme specified above. It corresponds to the intuition that using $\oplus$ and k random bits there are several ways to split a bit into shares *e.g.* a three way split of a bit b can be specified as $b \oplus r_1$, r_2, $r_1 \oplus r_2$ or as $b \oplus r_1 \oplus r_2$, r_1, r_2.

Let M be an adversary trying to distinguish the two distributions D_1 and D_2. It gets a sequence T of m samples, sampled from either D_1 or D_2, each element of which is a k tuple of signal values at the k points that the shares are accessed. If $S_1, \ldots, S_k$ are random variables denoting these values let $\mathcal{S} = (S_1 - \mu) \times \cdots \times (S_k - \mu)$ where μ is the mean of the distribution R. $\mathcal{S}$ has a slightly different mean (with the difference of $\frac{1}{2^{(k-1)}}$) under distributions D_1 and D_2 and with a variance of approximately $(\sigma^2)^k$. Using standard techniques, it is easy to show that an adversary given $(2\sigma^2)^k$ samples can distinguish the two distributions using the statistic $\mathcal{S}$. Thus, approximately n^k samples are sufficient, where $n = \sigma^2$. We are interested in lower bounds on the number of samples required to distinguish the distributions. Our central result is:

Theorem 4. *Let δ be a constant. Given distributions D_1 and D_2 defined above, any adversary which has access to $m < n^{\frac{k}{2}-4\delta}$ samples ($n = \sigma^2$) from one of these two distributions, has probability at most $n^{-\delta}$ of distinguishing D_1 and D_2.*

Note that this not a tight lower bound and we conjecture that n^k is the tight bound. We sketch the proof for the case $k = 2$ and the general proof can be done along the same lines. We require the following basic facts from probability theory.

3.5 Probability Theory Basics

The density function of *Normal distribution* with mean μ and variance σ^2 is

$$\eta(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}$$

The corresponding distribution function is defined as

$$N(x) = \int_{-\infty}^{x} \eta(x) dx$$

The following inequality is useful (see for example [5]):

$$N(x) < \frac{\sigma^2}{\mu - x} \eta(x)$$

Theorem 5. Chernoff Bound: *Let* $S_n = X_1 + ... + X_n$ *where the* X_i*s are independent and are* 1 *and* 0 *with probability* p *and* $q = 1 - p$*. For* $p < a < 1$*, and* $b = 1 - a$*,*

$$\Pr[S_n \geq na] \approx e^{-nK(a,p)}$$

where $K(a,p) = a\log(\frac{a}{p}) + b\log(\frac{b}{q})$.

3.6 Lower Bound for 2 way Split

In this section we outline a proof of Theorem 4 with $k = 2$. Our proof uses several techniques and ideas from Naor *et. al* [6]. The realizable distributions D_1 and D_2 which the adversary has to distinguish are defined on the space $\mathcal{P} = \mathcal{R} \times \mathcal{R}$, where $\mathcal{R}$ is the set of reals. If Y is the random variable sampled from one of these distributions, in D_1, $Y = \langle A + X_1, A + X_2 \rangle$ and $Y = \langle A + X_1, (1 - A) + X_2 \rangle$, where X_1, X_2 are random variables with normal distribution with parameters μ, σ and A is a uniform binary random variable.

Let M be an adversary trying to distinguish D_1 and D_2. By assumption, M fixes a certain precision ϵ and divides the area $\mathcal{R} \times \mathcal{R}$ into squares of length ϵ, where $\epsilon < \frac{1}{\sqrt{n}}$ without loss of generality. All inputs in a particular square are treated identically. When we refer to $\langle u, v \rangle \in \mathcal{R} \times \mathcal{R}$, we identify u and v with the boundaries of the intervals containing them and thus identify $\langle u, v \rangle$ with the the $\epsilon \times \epsilon$ square containing it.

Let $m = n^{1-4\cdot\delta}$, where δ is a constant. We show that no adversary can distinguish between sequences with at most m samples, sampled according to D_1 and D_2. In the following exposition, T is a random variable denoting a randomly drawn sequence, and s denotes a possible value of T. The outline of the proof is as follows: We first define a set of bad sequences (definition 6 below). Then we show (in Lemma 8) that under distributions D_1 and D_2 the probability that a sampled sequence T is bad i.e. the probability of the event BAD_T is very small. Restricting ourselves to sequences which are not bad, we show that the probability that the random variable T has a particular value s is almost the same whether we are sampling according to D_1 or D_2. In particular, in Lemma 10 we show that $\Pr_{D_1}(T = s | \neg BAD_T) > \mu_n * \Pr_{D_2}(T = s | \neg BAD_T)$, where μ_n is close to 1, from above. Similarly, we show that the probability of a sequence which is not bad, when sampled according to D_2 is at least μ_n^{-1} times its probability under D_1. In other words, the occurrence probability of a sequence that is not bad, is almost the same under both distributions. Putting it all together, we then show that the adversary cannot distinguish the distributions using fewer than m samples. We begin with the definition of bad sequences.

Definition 6. Let $f_s(x,y)$, $x, y \in \mathcal{R}$, be the number of times that $\langle x, y \rangle$ appears in sequence s. We call a sequence s a bad sequence if either (1) $f_s(\mu - u, \mu - v) > 0$, for $u, v > n^{(0.5+\delta)}$ or (2) $f_s(\mu - u, \mu - v) > \frac{n^2}{(uv+1)} \cdot n^{-c}$, for other values, $u, v > 0$. Here $c = 1 - 3\delta$.

In the above definition and in the rest of the proof we have ignored the cases when $u, v < 0$ and these can be treated symmetrically.

Definition 7. Define $maxf(\mu - u, \mu - v) = 0$, for $u, v > n^{(0.5+\delta)}$ and $maxf(\mu - u, \mu - v) = \frac{n^2}{(uv+1)} \cdot n^{-c}$, for all other values $u, v \geq 0$.

This the maximum possible number of times that $\langle u, v \rangle$ occurs in a sequence which is not bad. Denote the random sequence of m two tuples as $T = T_1 T_2 ... T_m$ and denote $s = s_1 s_2 ... s_m$.

Lemma 8. $\Pr_{D_1, D_2}(BAD_T) < e^{-n^\delta}$ i.e. *under either distribution, the set of bad sequences is negligibly small.*

Proof: We consider the two cases in the definition of a bad sequence separately. The probability that the random variable distributed according to $N(\mu, \sigma)$ takes on a particular value $\mu - x$ is given by $N(\mu - x) - N(\mu - x - \epsilon)$, which for small values of ϵ can be approximated by $\epsilon \cdot \eta(\mu - x)$. Using this approximation and taking into account the contribution of the binary valued random variable, under either distribution D_1 or D_2 the probability that $s_i = \langle \mu - u, \mu - v \rangle$ can be approximated by probability

$$p = \frac{d\epsilon^2}{(2\pi\sigma^2)} \cdot e^{-\frac{1}{2}(\frac{u^2+v^2}{\sigma^2})} \tag{2}$$

where d is a small constant close to 1. Since the elements of the sequence are sampled independently, by the Chernoff bound (section 3.5), the probability P_{uv} that $f_s(\mu - u, \mu - v) > \frac{n^2}{(uv)} \cdot n^{-c}$ is about $e^{-m \cdot K(a,p)}$, where $a = \frac{n^{-c}}{(uv+1)}$. Since $1 - a$ is close to 1, $K(a, p)$, a simple calculation shows that

$$P_{uv} \approx (\frac{p}{a})^{ma} \cdot (\frac{1-p}{1-a})^m < (\frac{d\epsilon^2 uvm}{2\pi\sigma^2 n^{2-c}})^{\frac{n^{2-c}}{uv}} \cdot e^{n^{1-c}} < (\frac{\epsilon^2 e}{n^{2\delta}})^{(\frac{n^{2-c}}{(uv+1)})}$$

Since there are $\frac{\sqrt{n^2}}{\epsilon^2}$ possible values of u, v, the total probability of BAD in case (2) is at most

$$\frac{n}{\epsilon^2} \cdot \epsilon^{2(\frac{n^{2-c}}{(uv+1)})}$$

which is exponentially small as $uv < n^{1+2\delta}$.

For case (1) of the definition of bad sequences, let s_i be the two tuple $\langle s_{i1}, s_{i2} \rangle$. For each i, using the inequality on Normal distribution in Section 2,

$$N(\mu - n^{0.5+\delta}) < \frac{\sigma^2}{n^{0.5+\delta}} e^{-\frac{1}{2}(\frac{n^{0.5+\delta}}{\sigma})^2}$$

Thus $\Pr(s_{i1} < \mu - n^{0.5+\epsilon(n)}) < e^{-(n^{2\delta} - \log n)}$. The probability that the sequence is bad according to case (1) is at most m times this small probability. □

Thus the space of bad sequences is very small. We now argue that for sequences that are not bad, the probability of occurrence is the same under both distributions. Denote $\Pr_{D_1}(T_i = \langle \mu - u, \mu - v \rangle)$, by $X_{u,v}$. Also, let $\Pr_{D_2}(T_i = \langle \mu - u, \mu - v \rangle)$ be $X_{u,v} + \Delta_{u,v}$. The difference $\Delta_{u,v}$ is due to the contribution of the binary valued random variable. The following lemma bounds $\Delta_{u,v}$.

Lemma 9. *For small* ϵ, $\frac{\Delta_{u,v}}{X_{u,v}} \leq \frac{uv}{\sigma^4}$.

Proof: This can be seen through the following sequence of approximations and identities. The first approximation follows from the definition of D_1 and D_2 and by choice of small ϵ.

$$\begin{aligned}
\Delta_{u,v} &\approx \frac{1}{2} \cdot (\eta(\mu-u)\eta(\mu-v) + \eta(\mu-u-1)\eta(\mu-v-1) \\
&\quad -\eta(\mu-u-1)\eta(\mu-v) - \eta(\mu-u)\eta(\mu-v-1)) \cdot \epsilon^2 \\
&= \frac{1}{2} \cdot \epsilon^2 \cdot (\eta(\mu-u) - \eta(\mu-u-1)) \cdot (\eta(\mu-v) - \eta(\mu-v-1)) \\
&= \frac{\epsilon^2}{2} \cdot \frac{1}{(\sqrt{2\pi}\sigma)^2} \cdot (e^{-\frac{1}{2}(\frac{u}{\sigma})^2} - e^{-\frac{1}{2}(\frac{u+1}{\sigma})^2}) \cdot (e^{-\frac{1}{2}(\frac{v}{\sigma})^2} - e^{-\frac{1}{2}(\frac{v+1}{\sigma})^2}) \\
&\leq \frac{\epsilon^2}{2} \cdot \frac{1}{2\pi\sigma^2} \cdot (\frac{u}{\sigma^2} \cdot e^{-\frac{1}{2}(\frac{u}{\sigma})^2}) \cdot (\frac{v}{\sigma^2} \cdot e^{-\frac{1}{2}(\frac{v}{\sigma})^2}) \\
&\leq \frac{d}{2} \cdot X_{u,v} \cdot \frac{uv}{\sigma^4}
\end{aligned}$$

The second last inequality follows from the power series expansion of e^x. The last inequality and the constant d are from (2). Thus the claim follows. □

Lemma 10. *The probability of occurrence of a sequence that is not bad is almost the same under both distributions. In particular,* $\mu_n^{-1} * \Pr_{D_1}(T = s | \neg BAD_T) < \Pr_{D_2}(T = s | \neg BAD_T) < \mu_n * \Pr_{D_1}(T = s | \neg BAD_T)$, *where* $\mu_n = 1 + (2n)^{-\delta}$.

Proof: We just show that

$$|Pr_{D_2}(T = s | \neg BAD_T) - Pr_{D_1}(T = s | \neg BAD_T)| < Pr_{D_1}(T = s | \neg BAD_T) * n^{-c\sqrt{n}}.$$

This follows by:

$$\begin{aligned}
&|Pr_{D_2}(T = s | \neg BAD_T) - Pr_{D_1}(T = s | \neg BAD_T)| \\
&= |\Pi_{u,v}(X_{u,v} + \Delta_{u,v})^{f_s(u,v)} - \Pi_{u,v}(X_{u,v})^{f_s(u,v)}| \\
&= \Pi_{u,v} X_{u,v}^{f_s(u,v)} \cdot |(\Pi_{u,v}(1 + \Delta_{u,v}/X_{u,v})^{f_s(u,v)}) - 1| \\
&\leq \Pi_{u,v} X_{u,v}^{f_s(u,v)} \cdot |(\Pi_{u,v}(1 + \Delta_{u,v}/X_{u,v})^{maxf(u,v)}) - 1| \\
&\leq \Pi_{u,v} X_{u,v}^{f_s(u,v)} \cdot |(\Pi_{u,v}(e^{n^{-c}})) - 1| \\
&\leq \Pi_{u,v} X_{u,v}^{f_s(u,v)} \cdot |((e^{n^{-c}})^m) - 1| \\
&\leq \Pi_{u,v} X_{u,v}^{f_s(u,v)} \cdot |(e^{n^{-\delta}}) - 1| \\
&\leq (\Pi_{u,v} X_{u,v}^{f_s(u,v)}) \cdot (2n)^{-\delta} \\
&= Pr_{D_1}(T = s | \neg BAD_T) \cdot (2n)^{-\delta}
\end{aligned}$$

The first two equalities are by definition 6 and the fact that $(1+x)^{1/x} < e$, for $x > 0$. Although the number of u, v pairs can be about (n/ϵ^2), the third inequality is true because the number of u, v for which $f_s(u, v) > 0$ is at most

m. The fifth inequality follows by the power series expansion of e^x, from which it can be shown that for $x < 1$, $e^x < 1 + 2x$. □

Proof (of Theorem 4) We put together the various pieces to show that an adversary M cannot distinguish these two distributions. Let M be a binary valued adversary and let $M(s)$ denote the output of M on input s, a sequence of samples. Note that if C is any condition on the random variables, by definition

$$Pr(M^D = 1 \mid C) = \Sigma_s Pr_D(T = s \mid C) \cdot Pr(M(s) = 1) \text{ and } T = s) \qquad (3)$$

By definition,

$$\begin{aligned}
&|Pr(M^{D_1} = 1) - Pr(M^{D_2} = 1)| = \\
&\quad |Pr(M^{D_1} = 1|\neg BAD_T) * Pr_{D_1}(\neg BAD_T) \\
&\quad - Pr(M^{D_2} = 1|\neg BAD_T) * Pr_{D_2}(\neg BAD_T) \\
&\quad + Pr(M^{D_1} = 1|BAD_T) * Pr_{D_1}(BAD_T) \\
&\quad - Pr(M^{D_2} = 1|BAD_T) * Pr_{D_2}(BAD_T)| \\
&\leq |Pr(M^{D_1} = 1|\neg BAD_T) * (Pr_{D_1}(\neg BAD_T) - \delta_n * Pr_{D_2}(\neg BAD_T))| \\
&\quad + |Pr(M^{D_1} = 1|BAD_T) * Pr_{D_1}(BAD_T) \\
&\quad - Pr(M^{D_2} = 1|BAD_T) * Pr_{D_2}(BAD_T)|
\end{aligned}$$

where $\mu_n^{-1} \leq \delta_n \leq \mu_n$. This follows from the observation (3) and Lemma 10. Since μ_n is close to one, and $\Pr_{D_1,D_2}(\neg BAD_T)$ is also close to one, the first summand on the right of the above inequality is close to zero. The second summand is also close to zero as $\Pr_{D_1,D_2}(BAD_T)$ is close to zero by lemma 8. Thus, the distinguishing probability is close to zero. □

Similarly we can show:

Theorem 11. *Let D_1 and D_2 be as before but with the noise being the sum of n identically and independently distributed binary variables (with $p = \frac{1}{2}$). Any adversary which has access to $n^{k/2-4\epsilon}$ samples, has probability at most $\frac{1}{n^\epsilon}$ of distinguishing D_1 and D_2.*

3.7 Encoding Bytes

For practical computation, we would use the encoding scheme of splitting each relevant byte of the computation into k shares. It is clear from our proof techniques that if there was enough additional noise in the power signals to effectively mask the byte values, then the same proofs will go through for the byte encoding scheme. It seems unlikely to happen in limited devices. It may be possible to extend our proof techniques to account for the fact that there is uncertainty on the value of a byte being manipulated given its power signal even without any additional noise.

4 Conclusions and Directions

We have presented a simplified initial step into the formal analysis of computing in the presence of loss of entropy due to leaked side channel information. Our lower bounds on the amount of side channel information required are proved for reasonable approximations of the actual distributions. Substantial effort is required to find more effective and general countermeasures against such attacks. Besides proving implementations secure from power attacks, this framework could also be used to design ciphers and other primitives which readily admit a secure, efficient implementation.

Acknowledgements

The authors thank D. Coppersmith, H. Scherzer, S. Weingart, M. Witzel and E. Yashchin for fruitful discussions about several issues presented in this paper.

References

1. P. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS and Other Systems. Advances in Cryptology-Crypto '96, Lecture Notes in Computer Science # 1109, pp 104–113.
2. P. Kocher, J. Jaffe and B. Jun. Differential Power Analysis: Leaking Secrets. In *Proceedings of Crypto '99*. A preliminary version is available online at http://www.cryptography.com/dpa/technical/index.html.
3. D. Chaum. Blind Signatures for Untraceable Payments. Advances in Cryptology: Proceedings of Crypto '82, Plenum Press, 1983, pp 199-203.
4. J. Daemen and V. Rijmen. Resistance against implementation attacks: A comparative study of the AES proposals. Proceedings of the Second AES Candidates Conference, Mar 1999, Rome, Italy.
5. W. Feller. An introduction to Probability Theory and its application, Vol. 1, Wiley Mathematical Statistics Series, 1950.
6. M. Naor, O. Reingold. On the construction of pseudo-random permutations: Luby-Rackoff revisited. In Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, pp 189–199.
7. M. Luby. Pseudorandomness and cryptographic applications. Princeton University Press.
8. The complete unofficial TEMPEST web page. Available at `http://www.eskimo.com/~joelm/tempest.html`.
9. M. Abadi, J. Feigenbaum, and J. Kilian. On Hiding Information from an Oracle. Journal of Computer and System Sciences, 39(1):21-50, Aug. 1989.

6 Appendix 1: Power distribution example

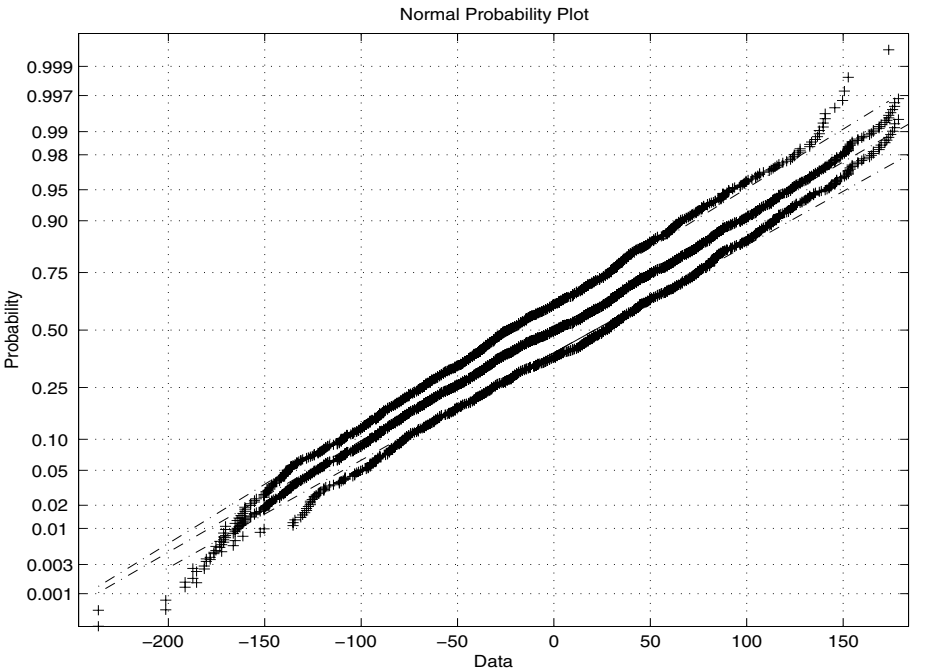

Fig. 1. Power distributions when loading random bytes from RAM

Figure 1 shows three distinct distributions of the instantaneous power consumption of a commonly available chip in the middle of a cycle which loads the value of a RAM byte into the accumulator. These correspond to three different distributions on the value of that particular RAM byte. All three power distributions are plotted on a "normal scale" and each distribution shows up as a thick line in this plot, which means all these three power distributions are close to normal. The middle line corresponds to the power distribution when the RAM byte is drawn uniformly at random. It has a mean of 0 (we have shifted all power readings by an additive constant to enforce this). The top line corresponds to the power distribution when the RAM byte is uniformly chosen from all bytes with MSB of 1. It has a mean of -25. The bottom line corresponds to the power distribution when the RAM byte is uniformly chosen from all bytes with MSB 0. This has a mean of $+25$.