Changing the odds against Masked Logic

Kris Tiri[1] and Patrick Schaumont[2]

[1] Trusted Platform Laboratory, Intel Corporation, USA,
kris.tiri@intel.com
[2] ECE Department, Virginia Tech, USA,
schaum@vt.edu

Abstract. Random switching logic (RSL) has been proposed as an efficient countermeasure to mitigate power analyses. The logic style equalizes the output transition probabilities using a random mask-bit. This manuscript, however, will show a successful attack against RSL. The single mask-bit can only add one bit of entropy to the information content of the overall power consumption variations and can very easily be deduced from the power consumption. Once the mask-bit is known, the a posteriori probabilities of the output transitions are not equal anymore and power analyses can be mounted. A threshold filter suffices to remove the additional bit of information.

1 Introduction

Side-channel attacks (SCAs) do not attack the mathematical properties of an encryption algorithm. Instead, they use information that is leaked by the device on which the algorithm has been implemented. Variations in power consumption, but also in time delay and electromagnetic radiation, have all successfully been exploited. In [7] for instance, a power attack extracts the full 128-bit key of an ASIC AES implementation in less then three minutes.

A SCA works as follows: it compares an estimation of the side-channel leakage with a measurement of the side-channel leakage. In a power-based SCA, measured power traces are compared with power consumption estimations. The correct key is found by identifying the best match between the measurements and the possible estimations. Furthermore, by limiting the side-channel leakage estimation to only a small piece of the algorithm, the computational complexity is reduced compared to a brute-force attack. A single AES secret key byte can be found by estimating the power consumption of only a single state register byte.

Countless SCA mitigations have been put forward. In case of a power attack, they range from decoupling the power supply or adding noise generators to masking data bearing signals or using custom logic cells. In [5] [6], Suzuki et al. combine the two ideas of masking data bearing signals and using custom logic cells into random switching logic (RSL). RSL unites the advantage of the former of being a theoretically proven countermeasure and the advantage of the latter of being an algorithmic independent countermeasure.

Yet as shown in [2] [3], theoretically proven mitigations do not always hold in practice. In this manuscript, we will successfully attack random switching logic. In fact, we will show that RSL only improves the power attack resistance by a factor of two.

Masking decorrelates the data from the power consumption. It equalizes the transition probabilities of the data bearing signals. If all signal transitions, i.e. 0 to 0, 0 to 1, 1 to 0 and 1 to 1, are equally likely and independent of the state of the circuit, a power analysis will be unsuccessful. In RSL, each and every signal is masked by xor-ing the output of all logic gates with a random mask-bit and consequently the output transition, and thus the power consumption, of the logic gate is independent of the state of the gate.

Masking is only effective if the mask-bit itself remains secret. If the value of this bit can somehow be derived from the measurements, the output transitions are not equally likely anymore. Therefore, knowledge of the mask-bit value will re-enable a normal power attack.

A single mask-bit can only impact the power consumption in a binary fashion. Even though all signal transitions are equally likely, the transition probability of a gate output is higher if the mask-bit changes. Indeed, the mask-bit can be seen as a data signal distributed to all gates. A change of its value will have an effect on all of the gates. It will result in a proportionally larger number of output transitions than when its value would remain constant. By filtering out the high power peaks, caused by these additional output transitions, we keep the events in which the mask-bit remains constant, thus in which the masking operation was actually absent.

The main contributions of this paper are that we point out that a single mask-bit only adds one bit of entropy, which is not sufficient to protect against power analyses and that we show how to successfully attack random switching logic. Additionally, compared with [3], we show that masking at the gate level can also be attacked when glitches are not present.

The remainder of this paper is organized as follows. The next section presents the information theory model of masking. It first introduces masking, probability and entropy and then applies it to RSL. In section 3, an experiment is setup in which, based of the findings of section 2, a test circuit implemented in RSL is successfully attacked. Finally a conclusion will be formulated.

2 Changing the odds using a posteriori probabilities

The power consumption of a CMOS circuit is dependent on architecture characteristics such as capacitance, supply voltage, leakage current and clock frequency. In addition, it is also dependent on signal activity. In the following, we will focus on the impact of this signal activity on information leakage. We will define some information theory concepts, and in particular analyze the entropy of masked signals. Using these concepts we will describe our proposed attack on RSL, which is discussed in the second subsection.

2.1 Conditional transition probability and entropy of random digital signals

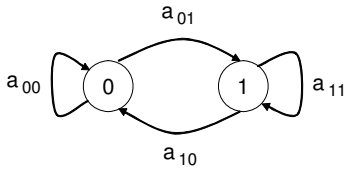
The dynamic power consumption of digital circuits is characterized in terms of the switching activity of the signals in the design. One defines the *activity factor* p_t of a signal a as the probability for a power-consuming transition per clock cycle of that signal [1]. A clock signal has $p_t = 1$. In static CMOS logic, data signals have a p_t smaller than 0.5. Often the activity factor is expressed as a percentage, and typical digital circuits show activity factors of 5% to 15% [9]. This means that a signal will show a power-consuming transition (i.e. 0→1 for CMOS) during 5% to 15% of the clock cycles.

For a signal a, we denote the probability of a zero as a_0 and of a one as a_1.

$$P(a=1) = a_1, \quad P(a=0) = a_0 = 1 - a_1 \tag{1}$$

We now establish a relation between p_t and the probability characteristics of a. We denote the absolute probability that a will make a transition from 0 to 1 as A_{01}. By our choice of p_t, it must be that $A_{01} = p_t$. We can now derive other relevant probabilities based on the probability a_0 and the activity factor p_t.

We will make use of a Markov model as illustrated in figure 1. In a Markov model, the random signal a is characterized in terms of its transition probabilities over a number of clock cycles. For each possible transition there is a corresponding probability defined as a_{ij}. For example, a_{01} is the probability that a becomes 1 under the condition that one clock cycle earlier it has the value 0. An important observation is that the conditional transition probabilities a_{ij} are bigger than absolute transition probabilities A_{ij}. In other words, the knowledge of the value of a influences our knowledge on the transitions made by a. This leads to the concept of *a posteriori probability*: the transition probability when the value of a is known.



$$A_{00} = a_0.a_{00}$$
$$A_{01} = a_0.a_{01}$$
$$A_{10} = a_1.a_{10}$$
$$A_{11} = a_1.a_{11}$$

Fig. 1. Markov Model for signal a

The long-term probabilities of the signal being zero (a_0) or one (a_1) are determined by the Markov chain in (2). This leads to the conclusion that random signals have an equal amount of up-going and down-going edges: $A_{01} = A_{10} = p_t$.

$$\left.\begin{array}{l}\begin{bmatrix} a_{11} & a_{01} \\ a_{10} & a_{00} \end{bmatrix}\begin{bmatrix} a_1 \\ a_0 \end{bmatrix} = \begin{bmatrix} a_1 \\ a_0 \end{bmatrix} \\ \Sigma a_i = \Sigma a_{1i} = \Sigma a_{0i} = 1 \end{array}\right\} \Rightarrow a_0.a_{01} = a_1.a_{10} \Rightarrow A_{01} = A_{10} \tag{2}$$

Entropy expresses the information content of a signal. For an event E with probability q, the entropy is equal to $H(E) = -\ q\ .\ \log_2(q)$. A random signal thus has a single bit of entropy:

$$H(a) = H(a_0) + H(a_1) = -½.\log_2(½) - ½.\log_2(½) = 1 \tag{3}$$

The entropy of the transitions of a is:

$$\begin{aligned} H(A) &= H(A_{00}) + H(A_{01}) + H(A_{10}) + H(A_{11}) \\ H(A) &= -p_t \log_2(p_t) - (a_0 - p_t)\log_2(a_0 - p_t) \\ &\quad -(a_1 - p_t)\log_2(a_1 - p_t) - p_t \log_2(p_t) \end{aligned} \tag{4}$$

When $a_0 = a_1 = 0.5$, the entropy of the transitions is between 1 and 2 bit, depending on the value of p_t.

$$H(A)\big|_{a_0=a_1=½} = -p_t(\log_2(p_t) - 1) - (1 - p_t)(\log_2(1 - p_t) - 1) \tag{5}$$

For $p_t = 0.5$, $H(A)$ reaches a maximum of 2 bits, as shown in figure 2. For other values in $0 < p_t < 1$, $H(A)$ is never smaller than 1 bit, even though $H(A)$ is derived from a single-bit random signal. Indeed, $H(A)$ is the result of observing two bits from a random stream. When these bits are uncorrelated (i.e. when the $p_t = 0.5$), we receive two bits of information per observation.

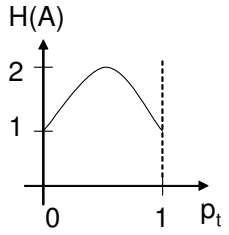


Fig. 2. Transition Entropy

We now examine the transfer of information through logic gates using the above concepts. Consider first a simple xor gate, with two signals at the input, one called r and another one called a. We will characterize the properties of the output signal q in terms of these two input signals.

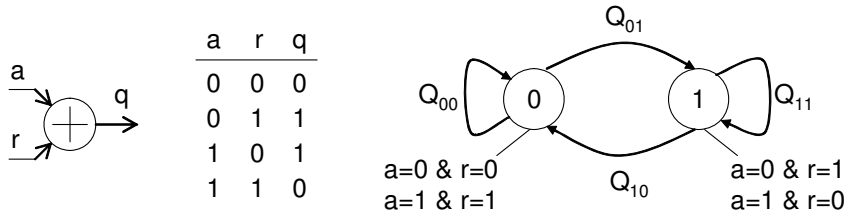


Fig. 3. Markov chain for an xor gate with inputs r and a

The transition probabilities of q can be expressed in terms of the transition probabilities of r and a. For example, the transition probability Q_{01} requires a and r to change from equal to different value. There are four combinations that have this effect, which results in Q_{01} having four terms.

$$\begin{aligned} Q_{01,xor} &= A_{00}R_{01} + A_{11}R_{10} + A_{10}R_{11} + A_{01}R_{00} \\ Q_{11,xor} &= A_{00}R_{11} + A_{11}R_{00} + A_{10}R_{01} + A_{01}R_{10} \\ Q_{10,xor} &= A_{00}R_{01} + A_{11}R_{01} + A_{10}R_{00} + A_{01}R_{11} \\ Q_{00,xor} &= A_{00}R_{00} + A_{11}R_{11} + A_{10}R_{10} + A_{01}R_{01} \end{aligned} \tag{6}$$

If r is an uncorrelated random signal (i.e. $R_{01} = R_{10} = R_{11} = R_{00} = 0.25$), then q will be an uncorrelated random signal as well. As a result, the entropy $H(Q_{xor})$ will be two bit, the same as $H(R)$. Only the xor gate has this property; an xor gate does not loose information and allows the signal a to be restored if signal r is known. In contrast, and and or gates destroy information, and their $H(Q)$ will be less than 2 bit. For example, assume an and gate with two input signals a and r, each with activity 50%, then it can be shown that

$$Q_{00,and} = 9/16, \quad Q_{01,and} = Q_{10,and} = 3/16, \quad Q_{11,and} = 1/16 \tag{7}$$

which leads to an entropy of $H(Q_{and}) = 1.66$ bit.

In the next section, we will apply the ideas of conditional probabilities and entropy to the observation of the power consumption of RSL gates. This will show that an apparently random signal can still have non-random a posteriori probabilities.

2.2 Random Switching Logic

The RSL nor and nand gates are defined as follows [5]:

$$nor_{rsl} : z = \overline{\bar{e} + x.y + (x+y).\bar{r}}$$
$$nand_{rsl} : z = \overline{\bar{e} + x.y + (x+y).r}$$

$$with \begin{cases} x = a \oplus r, \quad y = b \oplus r, \quad z = q \oplus r \\ q\big|_{nor_{rsl}} = \overline{a+b}, \quad q\big|_{nand_{rsl}} = \overline{a.b} \end{cases} \tag{8}$$

Signal r is the random mask-bit, which equalizes the transition probability according to formula 6. Signal e is an enable signal, which suppresses transient hazards. It prevents glitches which have been shown to make power analyses on masked gates possible [2]. The signal only enables the gate when input signals x, y and r are stable. For this purpose, signal e must meet stringent requirements such that for each gate its arrival time is later than the arrival times of the output of already enabled gates. For the rest of this manuscript, we will assume that the enable signal e is one and that glitches do not occur. Please note that for the experimental results, a cycle accurate simulator is used which does not simulate glitches.

In a design implemented with RSL, only the global input signals are explicitly masked. The internal nets are masked because of the RSL gates. A gate expects masked inputs and produces a masked output, which serves as the masked input of the next gate. Note that to implement this functionality, r is still distributed to and used by all gates.

The signal r modifies the functionality of the RSL gates as follows. When r equals zero, we can derive that:

$$nor_{rsl}\big|_{r=0} = \overline{x+y}$$
$$nand_{rsl}\big|_{r=0} = \overline{x.y} \tag{9}$$

and when r equals one:

$$nor_{rsl}\big|_{r=1} = \overline{x.y}$$
$$nand_{rsl}\big|_{r=1} = \overline{x+y} \tag{10}$$

Whenever the mask-bit value changes, each gate that previously functioned as a nand gate, modifies its functionality to a nor gate and vice versa. A design implemented with RSL will thus switch between two dual configurations, which will require energy in addition to the energy for calculating the design's output.

Table 1 presents the transition probabilities Z_{ij} of an RSL nand gate in function of the random mask-bit transition. The table has been calculated with the unmasked signals a and b having a typical activity factor of 10%. The summation per transition event of z shows that all transitions are equally likely (i.e. $Z_{01} = Z_{10} = Z_{11} = Z_{00} = 0.25$). This is the basis for the power attack resistance of RSL.

The table, however, also shows that the a prosteriori probabilities are not equally likely. For instance, if the mask-bit remains at one, with high probability the output z will remain at zero. Indeed, when r is one, the RSL nand gate functions as a nor gate, for which it is sufficient that one input is one to have a zero output.

Table 1. Transition Probabilities of an RSL nand gate with $A_{01} = 0.1$, $B_{01} = 0.1$.

r \ z	0 → 0	1 → 0	0 → 1	1 → 1
0 → 0	0.0400	0.0225	0.0225	0.1650
1 → 0	0.0225	0.0400	0.1650	0.0225
0 → 1	0.0225	0.1650	0.0400	0.0225
1 → 1	0.1650	0.0225	0.0225	0.0400
Σ	0.2500	0.2500	0.2500	0.2500

Table 1 shows that when the mask-bit r remains constant, it is very likely that the output z remains constant ($Z_{11} + Z_{00} >> Z_{01} + Z_{10}$ for r_{0-1} and r_{1-0}) while that when the mask-bit changes, it is very likely that the output changes ($Z_{01} + Z_{10} >> Z_{11} + Z_{00}$ for r_{0-1} and r_{1-0}). There will thus be a large difference in power consumption between the two events. This which will be easy to filter out, no matter what kind of design has been implemented. For example, if a typical design has a 10% activity factor, it means that the probability of a power transition of q would be 10% ($p_t = Q_{01} = Q_{10} = 0.1$) and it means also that the non-switching probability is 80% ($Q_{00} = Q_{11} = 0.4$). Now when r switches, these two groups are exchanged, and the circuit gets an activity factor of 40%.

Table 2 shows the transition probabilities after the constant mask-bit transitions have been selected. The a posteriori transition probabilities of the RSL nand gate are not equally likely anymore. They are asymmetric ($Z_{00} = Z_{11} = 0.41 \neq Z_{10} = Z_{01} = 0.09$) and a power attack should be possible.

Table 2. A posteriori transition probabilities of RSL nand gate

r \ z	0 → 0	1 → 0	0 → 1	1 → 1
0 → 0	0.0400	0.0225	0.0225	0.1650
1 → 1	0.1650	0.0225	0.0225	0.0400
2Σ	0.4100	0.0900	0.0900	0.4100

3 Experimental Results

3.1 Power Measurements

The power measurements are simulated with toggle counts. A toggle count reports how many signals have a switching event in a clock cycle. By restricting the toggle count to positive signal transitions, they report the power consuming transitions. This is a first order approximation of the power consumption. More accurate power consumption measurements can be obtained by using weight factors based on the estimated capacitance attached to the switching nets. For our purpose, raw un-weighted toggle counts are sufficient. If an implementation is not DPA proof with raw toggle counts it will not be DPA resistant with a more accurate model either. Please note that the inverse would not be true.

We obtained the toggle counts from our test circuit by simulation with the GEZEL cycle-based simulator (`http://rijndael.ece.vt.edu/gezel2`). This simulator supports two modes of toggle counting. In one mode, it obtains toggle counts from a test circuit for all intermediate nets as a function of time. In a second mode, it obtains toggle counts per net over all clock cycles. These counts are obtained by evaluating the Hamming distance of all signal transitions. The partial netlist, shown in figure 4, illustrates a circuit input description for this cycle-based simulator.

```
1. $option "profile_toggle_upedge_cycles" // count 0->1 transitions
2.
3.  ipblock rng(out q : ns(32)) {
4.    iptype "rngblock";
5.  }
6.
7.  dp rsl_nand(in  blank : ns(1);            // masking bit
8.              in  a, b  : ns(1);            // inputs
9.              out q     : ns(1)) {          // output
10.   always {
11.     q = ~((a & b) | ((a | b) & blank));
12.   }
13. }
```

Fig. 4. Circuit input description for GEZEL cycle-based simulator

Line 1 instructs the simulator to count up-going transitions in the circuit and to report the result per clock cycle.

Lines 3-5 create a random generator module by means of the ipblock construct. These ipblock are user-defined simulation primitives. They are described in C++, and are easy to add to the simulator. The advantage of such user-defined primitives for this application is that they do not contribute to the toggle count. Instead, ipblock primitives are black-box descriptions.

Lines 7-13 show the example model of an RSL-nand gate. This gate will be evaluated once per clock cycle during the simulation. The GEZEL simulator uses a pure cycle-based algorithm and does not simulate glitches. Besides the modeling of combinational logic, GEZEL also supports sequential logic, control modeling, and structural hierarchy. GEZEL also has a code generation backend to convert circuit descriptions into C++ as well as into synthesizable VHDL. The conversion into C++ is useful to generate ipblock descriptions automatically from existing circuit descriptions.

3.2 Device under test setup

Figure 5 shows the block diagram of the test circuit implemented in RSL. The test circuit consists of the AES substitution followed by the key addition. This is a sufficient subset of the AES algorithm on which a SCA can be mounted. Furthermore, a side-channel attack on AES will in general find the 128-bit secret key byte per byte by estimating the side-channel leakage of exactly the circuit shown in figure 5.

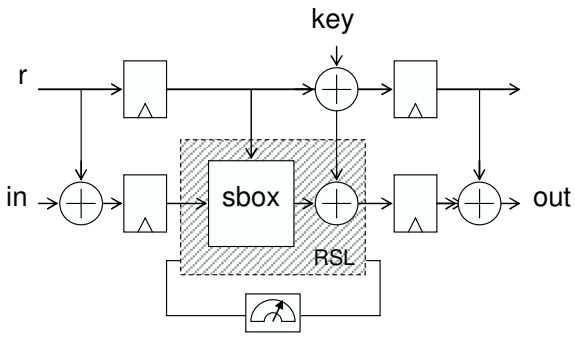


Fig. 5. RSL test circuit: AES sbox and key addition

The masking is done as follows. The random mask-bit r is inserted at the input, by xor-ing the input *in* and an 8-bit repetition of r. Likewise, the mask is inserted at the other input *key* and extracted at the output *out*. Additionally, the signal r is fed to the RSL gates forming the substitution box and the key addition.

The power measurements, i.e. the toggle counts, are restricted to the substitution box and the key addition. The switching of the registers storing signals r, *in*, and *out*, and the switching of the xor-gates masking signals *in*, *key* and *out* are not included in the power measurements. Only the toggle counts of the logic gates within the dotted line of figure 5 are reported. This has been done to exclude any side-channel leakage of unmasked signals from the measurements.

Additionally, even though the mask-bit r is a global signal distributed to all RSL gates it has only weight one. It has the same contribution into the toggle

count as a local signal confined between two adjacent gates. In reality, signal r behaves as a clock signal, which typically consumes a large fraction –30% to 40% according to [5]– of the total system power. The signal r, by itself, will thus cause a large power spike when switching. The observability of a mask-bit transition only increases with more accurate weight factors especially given that the signal r is distributed to all logic gates, 871 in our test circuit, while the clock is only distributed to the registers and latches, 18 in our test circuit.

The following power analysis has been carried out. The toggle count measurements are correlated with the number of changing bits between two subsequent values of the signal *in*. The number of changing bits serves as the attacker's estimate of the power consumption. The values of *in* are calculated from the signal *out*, which is known, and a guess on the signal *key*. The guess that results in the highest correlation coefficient is the correct secret key. Please note the power estimation is based on *in* and not on the actual state of the circuit $in \oplus r$.

3.3 Power based SCA results

The outcome of the power analysis is shown in figure 6. The results are based on 100,000 toggle count acquisitions, which is more than enough to disclose any side-channel leakage if present. In [7], less than 10,000 measurement acquisitions were required to extract the key in an actual measurement setup suffering from measurement errors and power dissipation of peripheral elements on the die.

The figure shows that the measurements and the estimations of all key hypotheses are uncorrelated. The correlation coefficients are very small and similar in size. Not one hypothesis really stands out and the attack did not expose the secret key. Based on these results, one could indeed conclude that RSL is successful in mitigating power analyses.

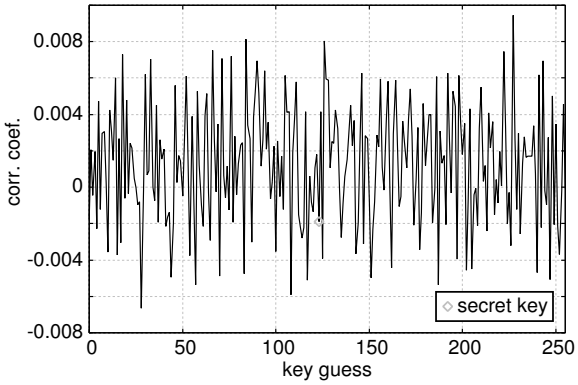


Fig. 6. Power side-channel analysis using 100,000 acquisitions

The mask-bit value, however, can be derived from a simple power analysis. Figure 7, which shows the toggle count transient together with the value of the

mask-bit for 100 clock cycles, confirms the derivations of section 2.2. The toggle count is higher than average whenever the mask-bit changes and smaller than average whenever the mask-bit remains constant.

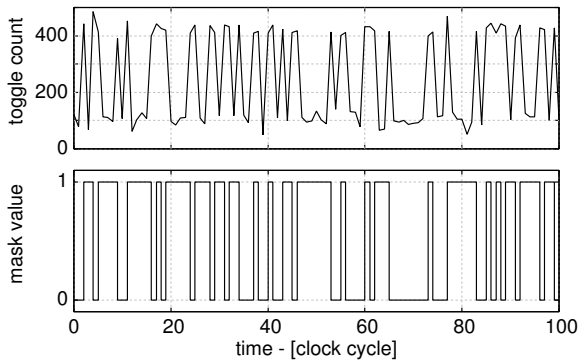


Fig. 7. Toggle count (top); and random mask-bit (bottom) transient.

The separation between the high and the low toggle counts corresponding to a switching and a non-switching mask-bit respectively is even more apparent in the toggle count histogram shown in figure 8. The figure shows that the single mask-bit only impacts the power consumption in a binary fashion. The entropy of the probability density function derived from the toggle count histogram of the test circuit fed with a constant mask-bit and a random mask-bit is 6.24 and 7.24 respectively. The single mask-bit adds exactly one bit of entropy to the information content of the overall power consumption variations.

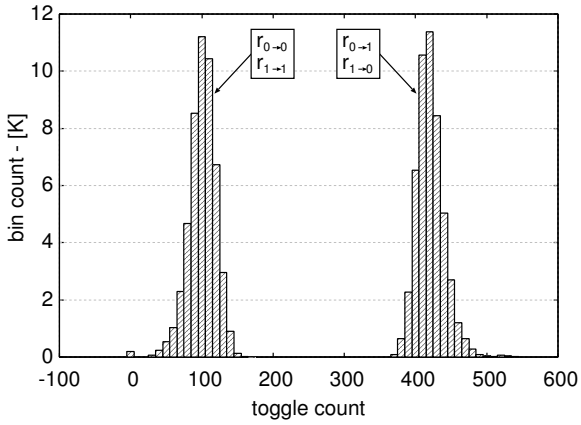


Fig. 8. Toggle count histogram based on 100,000 acquisitions.

Note that the toggle count values correspond closely with what would be obtained based on the transition probabilities of table 1. Based on the table, the test circuit, which consists of 388 nand gates and 483 nor gates, would have a toggle count of 380 whenever the mask-bit changes and a toggle count of 80 whenever the mask-bit remains constant.

The toggle count numbers also confirm that gate level masking is an expensive operation. Recall that whenever the mask-bit changes, the configuration switches to its dual mode. Each gate that previously functioned as a nand gate, changes it functionality to a nor gate and vice versa. About 3 times the normal toggle count is required to switch between both configurations. The un-weighted toggle count power model, which essentially neglects the power dissipation of the global masking signal r and the global enable signal e, estimates the mean power consumption penalty of RSL to 150% when compared with a regular implementation.

A threshold filter, which filters out the large toggle counts, retains the events of interest while at same time throws away the unwanted events. The operation only keeps the events in which the mask-bit remains constant, thus in which the masking operation was absent. The remaining toggle counts can come from the test circuit in a stable nand-nor configuration or from the test circuit in a stable nor-nand configuration. The fact that these are two different configurations is not important. It is sufficient for the power analysis that both have a power consumption profile proportional to the Hamming distance of two subsequent states. They do not need to have exactly the same power consumption profile.

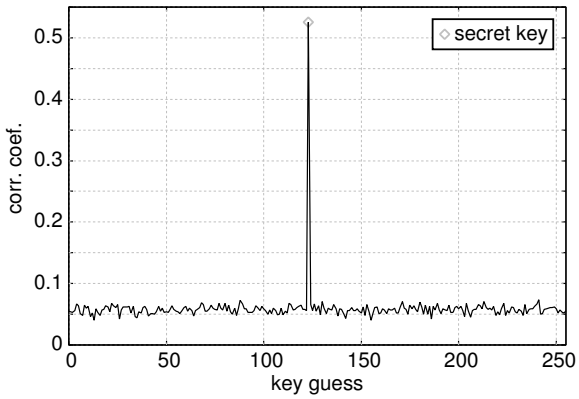


Fig. 9. Power side-channel analysis after threshold filtering operation on 100,000 acquisitions.

The outcome of the power analysis after a threshold filter has removed the samples with a toggle count larger than 250 is shown in figure 9. The power attack successfully exposes the secret key. In fact, 100 acquisitions (yielding approximately 50 samples after the filter) are sufficient to disclose the correct secret

key. These experimental results show that RSL is not an efficient countermeasure to mitigate power analyses.

4 Conclusions

Masking is only effective if the mask-bits remain secret. Once the mask-bits are known, the output transitions are not random anymore. A single mask-bit can only impact the power consumption in a binary fashion. For RSL, the mask-bit can easily be derived from the power measurements. A switching mask-bit causes the energy consumption of a typical design to increase four-fold. Once the low energy counts have been separated from the high energy counts, random switching logic can successfully be attacked.

References

1. Chandrakasan, A., Sheng, S., and Brodersen, R.: Low Power CMOS Design. IEEE Journal of Solid-State Circuits (JSSC), 27(4):473–484, April 1992.
2. Chandrakasan, A., Sheng, S., and Brodersen, R.: Side-Channel Leakage of Masked CMOS Gates. The Cryptographers' Track at the RSA Conference (CT-RSA 2005), Lecture Notes in Computer Science, 3376:351–365, February 2005.
3. Peeters, E., Standaert, F., Donckers, N., Quisquater, J.: Improved Higher Order Side-Channel Attacks with FPGA experiments. Cryptographic Hardware and Embedded Systems (CHES 2005), Lecture Notes in Computer Science, 3659:309–323, August 2005.
4. Moyer, B.: Low-power design for embedded processors. Proceedings of the IEEE, 89(11):1576–1587, November 2001.
5. Suzuki, D., Saeki, M., and Ichikawa, T.: Random Switching Logic: A Countermeasure against DPA based on Transition Probability. Cryptology ePrint Archive, Report 2004/346, 2004.
6. Suzuki, D., Saeki, M., and Ichikawa, T.: DPA Leakage Models for CMOS Logic Circuits Cryptographic Hardware and Embedded Systems (CHES 2005), Lecture Notes in Computer Science, 3659:366–382, August 2005.
7. Tiri, K., Hwang, D., Hodjat, A., Lai, B., Yang, S., Schaumont, P., and Verbauwhede, I.: Prototype IC with WDDL and Differential Routing - DPA Resistance Assessment. Workshop on Cryptographic Hardware and Embedded Systems (CHES 2005), Lecture Notes in Computer Science, 3659:354–365, August 2005.
8. Weste, N., and Harris, D.: Principles of CMOS VLSI Design. Addison-Wesley, third edition, 2005.