Power Analysis Attacks and Countermeasures

Thomas Popp
Graz University of Technology

Elisabeth Oswald
University of Bristol

Stefan Mangard
Infineon Technologies

Editor's note:
Side-channel attacks represent a threat to most systems that contain cryptographic implementations, completely bypassing the theoretical strength of the underlying algorithms. This article presents an overview of power analysis attacks, which are based on the measurement of the power consumed by cryptographic ICs, and countermeasures against them.
—Anand Raghunathan, NEC Laboratories America

THE GOAL OF attacks on cryptographic devices is to reveal secret information that is stored inside them. Although all such attacks have essentially the same objective, they differ significantly in terms of required cost, time, equipment, and expertise. One of the biggest challenges for designers of cryptographic devices is to protect the devices from implementation attacks. In contrast to classical cryptanalysis attacks, which target the employed cryptographic algorithms' mathematical weaknesses, implementation attacks aim at the properties of the actual circuits and chips implementing these algorithms.

In particular, passive, noninvasive attacks (where attackers observe a device's physical properties and only exploit its accessible interfaces) have received a lot of attention and scrutiny. These attacks are also called *side-channel attacks*. (See the "Categorizing implementation attacks" sidebar for a full definition of each type of attack on cryptographic devices.) The three most important types are timing, power-analysis, and electromagnetic (EM) attacks. In these cases, the basic idea is to determine a cryptographic device's secret key by measuring its execution time, power consumption, or electromagnetic field.

This article focuses on power analysis attacks[1] because they have received by far the most attention in recent years. They are powerful and can be executed relatively easily. This article provides an introduction to these attacks and discusses countermeasures against them. In particular, we focus on countermeasures that can be implemented at the cell level. (A comprehensive discussion of all kinds of power analysis attacks and countermeasures is available elsewhere.[2])

Power analysis attacks

In power analysis attacks, the attacker attempts to reveal secret information that is stored inside the device, on the basis of the cryptographic device's power consumption. This targeted information is typically a secret key that is used for a cryptographic algorithm, so we refer to this information as the secret key for the remainder of this article.

In practice, an attacker can access a cryptographic device in various ways. For instance, the attacker usually knows the device's inputs or outputs (such as plaintexts or ciphertexts). Moreover, the attacker can typically invoke the execution of a certain cryptographic algorithm using a certain (but unknown) secret key. For power analysis attacks, the attacker also needs a model of the cryptographic device. This model can be crude if little is known about the cryptographic device, or it can be sophisticated if the attacker has, for instance, full access to all specifications or even low-level descriptions (netlists) of the device. Attackers use the model to predict certain intermediate values that they assume will occur in the cryptographic algorithm's computation. These inter-

0740-7475/07/$25.00 © 2007 IEEE Copublished by the IEEE CS and the IEEE CASS

Categorizing implementation attacks

Even though all implementation attacks on cryptographic devices have essentially the same goal, their requisite cost, time, equipment, and expertise can differ significantly. We can look at whether they are passive or active and their degree of invasiveness.

In the first category, a *passive attack*, the cryptographic device is operated largely, or even entirely, within its specification. Attackers reveal secret information by observing the device's physical properties (such as execution time and power consumption). In an *active attack*, the cryptographic device, its inputs, and/or its environment are manipulated to make the device behave abnormally. The secret key is revealed by exploiting the device's abnormal behavior.

The second category involves looking at the interface that the attack exploits. Cryptographic devices have several physical and logical interfaces. Some of these interfaces can be accessed easily, whereas others can be accessed only with special equipment. Based on the interface that an attack uses, we can distinguish between invasive, semi-invasive, and noninvasive attacks. All these attacks can be passive or active:

- Invasive attacks are the strongest type that can be mounted on a cryptographic device. In such an attack, there are essentially no limits to what is done with the cryptographic device to reveal secret information. For example, the attacker might probe and manipulate the device's wires using a focused ion beam (FIB). Invasive attacks are extremely powerful, but they typically require expensive equipment. Consequently, only few publications exist on this topic.[1]
- Semi-invasive attacks involve depackaging the cryptographic device. However, in contrast to invasive attacks, no direct electrical contact with the chip surface is made—the passivation layer stays intact. Such attacks typically do not require as expensive equipment as invasive ones. An example of a semi-invasive attack is fault induction by light. (Skorobogatov's doctoral dissertation is the most comprehensive publication on semi-invasive attacks.[2])
- In noninvasive attacks, the cryptographic device is essentially attacked as is; attackers only exploit directly accessible interfaces. The device is not permanently altered, so no evidence of an attack is left behind. Most noninvasive attacks can be conducted with relatively inexpensive equipment, so these attacks pose a serious practical threat to the security of all kinds of cryptographic devices.

Different types of implementation attacks require different countermeasures that vary significantly in strength and incurred overheads. Which implementation attacks designers of a cryptographic device have to anticipate depends mainly on the value that is protected by a particular device.

References

1. O. Kömmerling and M.G. Kuhn. "Design Principles for Tamper-Resistant Smartcard Processors," *Proc. Usenix Workshop Smartcard Technology* (Smartcard 99), Usenix Assoc., 1999, pp. 9-20.
2. S.P. Skorobogatov, "Semi-Invasive Attacks: A New Approach to Hardware Security Analysis," doctoral dissertation, Computer Lab., Univ. of Cambridge, 2005; http://www.cl.cam.ac.uk/TechReports.

mediate values depend on the known input or output values and also on the unknown secret key. Hence, attackers must guess a part of the secret key.

To reveal this part of the secret key, attackers map the intermediate values to hypothetical power consumption values. They then compare their hypothetical power consumption values to the device's real power consumption, which they measured beforehand when processing the known input or output values. If all their assumptions (the cryptographic device's model and power consumption characteristic) and their key guess were correct, then the hypothetical power consumption values correlate with the real values. If this correlation is high enough, then they accept the key guess as correct. If there is little correlation, then one or several of their assumptions were incorrect.

Clearly, power analysis attacks involve many assumptions. In particular, the mapping from intermediate values to hypothetical power consumption values is critical. In many publications, researchers use standard power consumption models, such as the bit model (which value has a bit?), the Hamming-weight model (how many bits of a value are 1?), and the Hamming-distance model (how many bits of a value change its state?). Such models often work

well if intermediate values that are stored in registers are attacked, but they do not work well in many other cases.

In other instances, the attacker might have detailed knowledge about a device's power consumption characteristics. They could obtain such knowledge, for example, by characterizing the device before the actual attack. Power analysis attacks that use such a characterization are called *template attacks*.[3] In this article, we focus on attacks in which the attacker uses standard power models. Depending on how the analysis of the power traces proceeds, we can distinguish between simple power analysis (SPA) and differential power analysis (DPA) attacks.

SPA attacks

The goal of SPA attacks is to reveal the secret key when given only a few power traces (that is, for a small number of input or output values). In the most extreme case, this means that the attacker attempts to reveal the key on the basis of only a single power trace. The attacker must be able to monitor the power consumption of the device under attack. In the attacked device, the secret key must have (directly or indirectly) a significant impact on the power consumption. SPA attacks exploit key-dependent differences (patterns) within a trace. Attackers typically identify these patterns by visually inspecting the recorded power traces.

SPA attacks are useful in practice if only one or a few traces are available for a given set of inputs. Consider, for example, a scenario in which a customer uses a smart card to pay at a gas station. The customer must regularly refill the car's gas tank and always buys a similar amount of gas. A malicious smart-card reader could record the card's power consumption. With this, attackers could gather a couple of traces for similar plaintexts. These few traces could then let the attackers learn the smart-card personal identification number (PIN).

DPA attacks

In contrast, DPA attacks require many power traces. Therefore, it is usually necessary to physically possess a cryptographic device for some time to mount a DPA attack on it. For example, with an electronic purse, someone could record numerous power traces by transferring small amounts of money to and from the purse. The attacker could then analyze these traces to reveal the secret key that the purse uses to protect the stored cash.

Another important difference between the two kinds of attacks is that the recorded traces are analyzed in different ways. In SPA attacks, a device's power consumption is analyzed mainly along the time axis. The attacker tries to find patterns in a single trace. In DPA attacks, the shape of the traces along the time axis is not as important. DPA attacks analyze how the power consumption at fixed moments of time depends on the processed data.

Example

We can demonstrate a DPA attack to show how it reveals the first byte of the secret key of an Advanced Encryption Standard (AES) software implementation.[4] The software implementation is executed on an 8051 microcontroller, and the power consumption is recorded using a digital-sampling oscilloscope. The microcontroller receives plaintexts via an RS-232 interface, encrypts them, and returns the corresponding ciphertexts. When attacking the device, we therefore have access to the plaintexts and ciphertexts.

To reveal the first byte of the key, we exploit the fact that the microcontroller's power consumption at some moment in time depends on the output of the first AES substitution box (S-box) operation in round one. This intermediate result of the algorithm is a function of the first byte of plaintext and the first byte of the secret key. After choosing this intermediate result, we record the microcontroller's power consumption during the first round of AES while it encrypts 1,000 different plaintexts. We store the power traces in a matrix $\boldsymbol{T}$ of power consumption values, where each row represents one power trace.

In the next step, we calculate hypothetical intermediate values for the first S-box operation in each of the 1,000 encryption runs. This means we calculate the values $v_{i,j} = \text{S-box}(d_i \oplus k_j)$, where $d_1, \ldots, d_{1,000}$ represent the first byte of each of the 1,000 plaintexts, and $k_j = j - 1$ with $j = 1, \ldots, 256$ constitute the possible values of the key's first byte. The matrix $\boldsymbol{V} = (v_{i,j})$ then has a size of $1{,}000 \times 256$ values.

The fourth step of the DPA attack is to map $\boldsymbol{V}$ to a matrix $\boldsymbol{H} = (h_{i,j})$ of hypothetical power consumption values. In the current attack, we use the simple bit model for this mapping. We simply consider the least significant bit (LSB) of the values in $\boldsymbol{V}$. Hence, we use $h_{i,j} = \text{LSB}(v_{i,j})$ as our power model. On the basis of $\boldsymbol{H}$, we can then perform the DPA attack's last step. We calculate the correlation coefficients between every column of $\boldsymbol{H}$ and all columns of the recorded power

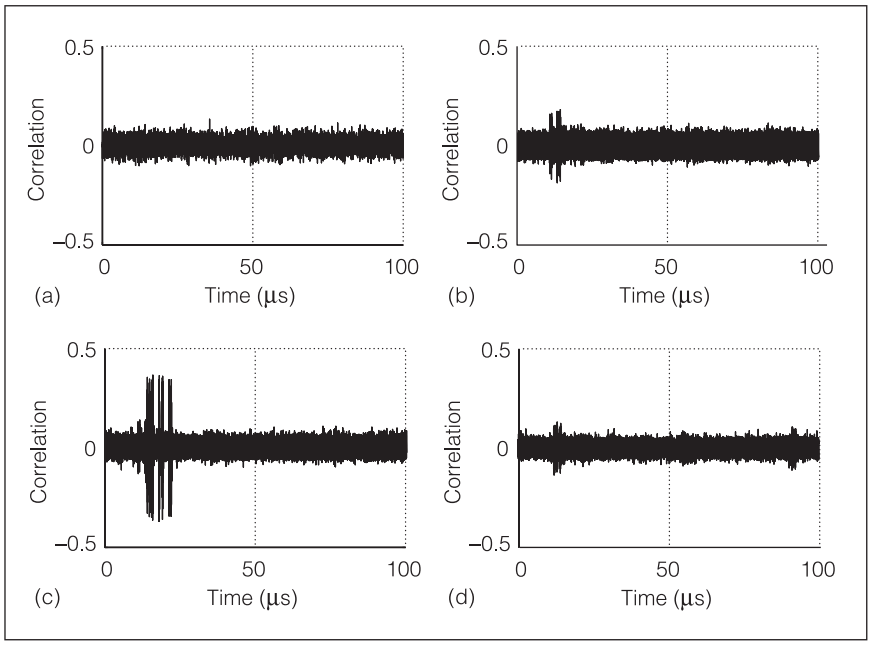

Figure 1. Result of a successful differential power analysis (DPA) attack, showing correlation traces for key hypotheses 223 (a), 224 (b), 225 (c), and 226 (d). The significant higher correlation peaks for key hypothesis 225 (c) compared to the other hypotheses clearly tells the attacker that the first byte of the AES key is 225. (Source: Mangard, Oswald, and Popp,[2] with kind permission of Springer Science and Business Media.)

consumption values ***T***. The result of this calculation is a matrix ***R*** of correlation coefficients.

Each row of ***R*** corresponds to one key hypothesis—that is, it has been calculated with a particular column of ***H***. Figure 1 shows the rows for this attack's key hypotheses 223 to 226. We can see there are high peaks in the plot for key hypothesis 225. In fact, these peaks are the highest ones in the entire matrix ***R***; all other values of ***R*** are considerably smaller, indicating that the first byte of the microcontroller's secret key is 225.

Countermeasures

Power analysis attacks work because the cryptographic devices' power consumption depends on the executed cryptographic algorithms' intermediate values. Hence, the goal of countermeasures against such attacks is to make the power consumption independent of those intermediate values. To date, protocol, hiding, and masking countermeasures are the basic groups into which these countermeasures can be characterized.

Protocol

Keys that are only used for a few cryptographic operations are usually called *session keys*. Using session keys can make power analysis attacks considerably more difficult. Obviously, the fewer traces attackers can obtain for a fixed key, the less information they can retrieve. Hence, session keys should be used whenever possible.

However, there are many scenarios when it is impractical or impossible to update secret keys frequently enough to prevent power analysis attacks. Therefore, hiding and masking countermeasures are generally the first line of defense for cryptographic devices.

Hiding

Hiding helps remove the power consumption's data dependency. This means the device's power consumption characteristics are changed so that an attacker cannot easily find a data dependency. Designers can change the power consumption by building the device such that every operation requires approximately the same amount of energy or such that the power consumption is more or less random. Both cases significantly reduce the data dependency of the power consumption.

It is important to point out that implementations protected by hiding countermeasures process the same intermediate results as unprotected implementations. Resistance against power analysis attacks is achieved solely by altering the cryptographic device's power consumption characteristics.

Masking

The idea behind masking is to randomize the intermediate values that the cryptographic device processes. The motivation behind this approach is that the power that is needed to process randomized intermediate values is largely independent of the actual intermediate values. An advantage of masking is that the device's power consumption characteristics do not need to be changed.

Countermeasures at the cell level

The principles of the countermeasures we've introduced so far can be implemented at different levels in a cryptographic device. For example, it is

possible to include mechanisms to hide the power consumption in software, the hardware architecture, or the cells that are used to implement the hardware. In this article, we provide a brief overview of countermeasures at the cell level.

The basic idea here is to build cryptographic circuits out of cells that are resistant to power analysis attacks; that is, their power consumption is independent of the intermediate values of the executed cryptographic algorithms. Since a circuit's power consumption is the sum of the power consumption of its cells, the circuit will be resistant to power analysis attacks if its cells are not (de)activated in a data-dependent manner (for example, by using data-dependent clock gating).

At the cell level, using hiding and employing masked logic styles can help designers build up power-analysis-resistant cells. In the case of hiding, by making a cell's power consumption identical in every clock cycle, its power consumption also becomes independent of the processed data values. If a masked logic style is used, the cells process only randomized values and the corresponding masks, making their power consumption largely independent of the executed algorithms' intermediate values. (Further information on these logic styles is available elsewhere.[2])

Hiding logic styles

The first structured approach to counteract power analysis attacks at the cell level was the use of hiding logic styles. These styles try to break the correlation between an algorithm's intermediate results and the power consumption of the cryptographic device that executes this algorithm by making the instantaneous power consumption of the cells the same in each clock cycle. As a result, the device's power consumption is identical in each clock cycle and has a maximum value in each clock cycle, which is one of the drawbacks of using this approach. Still, hiding logic styles counteract both SPA and DPA attacks.

The three major types of hiding logic styles are dual-rail precharge, asynchronous, and current-mode logic styles. *Dual-rail precharge* (DRP) logic styles are the most popular type. As the name implies, the concepts of dual-rail and precharge logic are combined to achieve a constant power consumption. Dual rail means that every signal s_i is encoded in a differential manner on two complementary wires, w_i and $\overline{w_i}$: $s_i = 0 => w_i = 0, \overline{w_i} = 1$; and $s_i = 1 => w_i = 1, \overline{w_i} = 0$. Thus, this approach conceals a particular signal's actual value. Precharging breaks a signal's sequence of values by splitting each clock cycle into precharge and evaluation phases. In the precharge phase, the complementary wires encoding a signal are set to a predefined precharge value, such as 0. In the subsequent evaluation phase, one of the two complementary wires is set to 1 according to the actual value that is processed. As a result, for each signal in a circuit, exactly one $0 \rightarrow 1$ transition and one $1 \rightarrow 0$ transition occur in a clock cycle. By ensuring a balance between the complementary wires between cells on the one hand and a balance of the internal structure of the cells on the other hand, designers can achieve a constant power consumption. Examples of DRP logic styles are sense-amplifier-based logic (SABL), wave-dynamic-differential logic (WDDL), dual-spacer dual-rail logic (DSDR), three-phase dual-rail precharge logic (TDPL), and three-state dynamic logic (3sDL).

The second type of hiding logic style is asynchronous logic. You might assume that the asynchronous behavior of circuits built using such logic styles helps to increase their power analysis resistance because it acts like some kind of randomization. However, this kind of randomization is inherently data dependent and thus does not lead to high security. As a result, asynchronous logic styles typically resort to the DRP principle to achieve a constant power consumption. Various researchers have studied the applicability of asynchronous logic for power-analysis-resistant circuits and have produced a test chip that includes a DPA-resistant, asynchronous microcontroller.[5]

The third and last type of hiding logic style is current-mode logic (CML), in which logic values are not encoded by different voltage levels but by current flows that take different paths in a circuit. Since these current flows are rather constant, the power consumption of CML circuits is also constant. This makes CML styles an interesting choice for power-analysis-resistant circuits. Examples of proposed CML styles for such circuits are metal-oxide semiconductor current-mode logic (MCML) and dynamic current-mode logic (DyCML).

As an example of a hiding logic style, we can look in more detail at the DRP logic style SABL.[6] The specific design of SABL cells ensures that their internal power consumption is constant and that their time of evaluation (TOE) is data independent; that is, the cells evaluate (calculate their output values as a function of the input values) only after all input signals have been set to complementary values. Because of this design, SABL cells are highly resistant to power analysis but

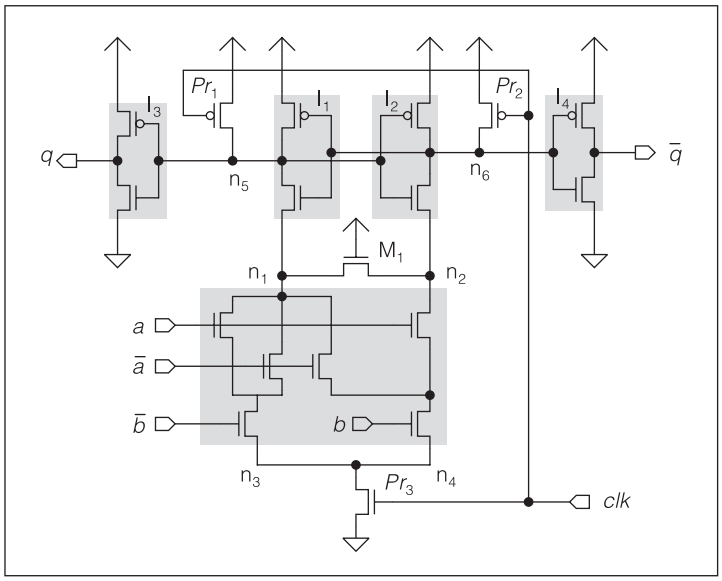

Figure 2. Schematic of the DPA-resistant sense-amplifier-based logic (SABL) NAND cell. This logic style uses hiding as a DPA countermeasure. That is, the power consumption of the cell is identical for all possible input and output values. (Source: Mangard, Oswald, and Popp,[2] with kind permission of Springer Science and Business Media.)

must be implemented from scratch. In SABL circuits, the combinational cells are connected to the clock signal, and all cells precharge simultaneously.

Figure 2 shows the transistor schematic of an SABL NAND cell. The network of NMOS transistors controlled by the input signals a, $\bar{a}$, b, and $\bar{b}$ defines the cell's functionality and is called a differential pull-down network (DPDN). During the precharge phase, the clock signal *clk* is 0, the precharging NMOS transistor Pr_3 is off, and all the cell's internal nodes are charged to 1 via the precharging PMOS transistors Pr_1 and Pr_2. The output inverters I_3 and I_4 ensure that the output signals q and $\bar{q}$ are precharged to 0. In the subsequent evaluation phase, first the clock signal is set to 1. This causes Pr_1 and Pr_2 to insulate and Pr_3 to conduct. However, because the inputs are still at the precharge value 0, the DPDN is not yet conducting. When the inputs are finally set to complementary values, the last arriving input establishes a conducting path from node n_3 or n_4 to node n_1 or n_2 and sets either n_3 or n_4 to 0. As a result, the corresponding inverter I_1 or I_2 is activated and the respective output node n_5 or n_6 is set to 0. Because I_1 and I_2 are connected in a cross-coupled fashion, the opposite node stays at 1. Setting either node n_5 or n_6 to 0 causes exactly one output q or $\bar{q}$ to switch to 1. The minimally sized NMOS transistor M_1 acts as a resistor and ensures that all DPDN nodes are data-independently discharged to 0 in the evaluation phase. In the subsequent precharge phase, all the cell's internal nodes that have been discharged to 0 are charged again. Considering the SABL cell's functionality, to maintain the cell's constant internal power consumption, the nodes n_5 and n_6 must be balanced. SABL flip-flops consist of two SABL latches that are contrariwise set (that is, when the latch at the input is set to the precharge phase, the latch at the output is set to the evaluation phase, and vice versa) to the precharge and evaluation phases.

Masked logic styles

Masking at the cell level has become popular during the past few years. Before, masking was mainly used at the architecture level. As a result, only a few practical results are available for this type of cell-level countermeasure. In more recent research projects, such as the Side-Channel Analysis Resistant Design Flow (Scard) project (http://www.scard-project.eu), researchers have analyzed masked logic styles in more detail.

Using a masked logic style, designers also break the correlation between an algorithm's intermediate values and the power consumption of the cryptographic device that executes this algorithm. Each intermediate value v is masked by a random value m. The cells then process only the masked intermediate value v_m and the corresponding mask m. Because the unmasked value v and the masked value v_m are uncorrelated, the cells' power consumption also remains uncorrelated to v. Designers typically use the XOR function as the masking operation, or $v_m = v \oplus m$, which is called Boolean masking. Another type is called arithmetic masking, which uses an arithmetic operation to combine v and m. If the masked cells are not activated in a data- or operation-dependent manner, masked logic styles counteract both SPA and DPA attacks.

Examples of masked logic styles are masked dual-rail precharge logic (MDPL),[7] random switching logic (RSL), and dual-rail random switching logic (DRSL). These logic styles use a single mask per intermediate value and circuit. In a more general approach called secret sharing, several masks are applied to an intermediate value. In that sense, masking is a secret sharing scheme based on the two shares v_m and m.

Table 1. Truth table of an MDPL NAND cell for complementary input values.

a	b	q	m	a_m	b_m	q_m	$\overline{m}$	$\overline{a}_m$	$\overline{b}_m$	$\overline{q}_m$
0	0	1	0	0	0	1	1	1	1	0
0	0	1	1	1	1	0	0	0	0	1
0	1	1	0	0	1	1	1	1	0	0
0	1	1	1	1	0	0	0	0	1	1
1	0	1	0	1	0	1	1	0	1	0
1	0	1	1	0	1	0	0	1	0	1
1	1	0	0	1	1	0	1	0	0	1
1	1	0	1	0	0	1	0	1	1	0

As an example of a masked logic style, we look in more detail at MDPL. In an MDPL circuit, all values v are concealed by a single mask m using Boolean masking: $v_m = v \oplus m$. The circuit is also implemented as a DRP circuit to avoid glitches that make masked circuits vulnerable to power analysis attacks.[8] However, the complementary wires in MDPL circuits do not need to be balanced. Designers can implement MDPL cells using commonly available single-rail (SR) standard cells. Only sequential cells are connected to the clock signal; the combinational cells precharge their outputs when their inputs have been set to the precharge value. Thus, the precharge value moves like a wave through the combinational logic blocks in the precharge phase.

Table 1 shows the truth table of an MDPL NAND cell for complementary input signals. The table demonstrates that we can calculate an MDPL NAND cell's output signals q_m and $\overline{q}_m$ using a majority (MAJ) function from the input signals in this way: $q_m = \mathrm{MAJ}(\overline{a}_m, \overline{b}_m, \overline{m})$ and $\overline{q}_m = \mathrm{MAJ}(a_m, b_m, m)$. A majority function produces 1 as its output if more inputs are set to 1 than to 0. Fortunately, a MAJ cell is available in typical single-rail, standard-cell libraries. Figure 3 shows the cell schematic of an MDPL NAND cell implemented using MAJ cells. Other MDPL cells implementing AND, (N)OR, and X(N)OR functionality have a similar structure.

MDPL flip-flops must carry out two additional tasks besides storing masked data values. They start the precharge wave that moves through the combinational logic blocks, and they change the mask value $m(t)$ of one clock cycle to the mask value $m(t + 1)$ of the next clock cycle.

Limits of DPA-resistant logic styles

At first glance, DPA-resistant logic styles seem to solve the problem of power analysis attacks. However, there are limits to their security. Here, we briefly introduce the main issues that designers of DPA-resistant logic styles and circuits must consider:

First, a major issue of hiding logic styles is the *balancing* of the cell and interconnect layouts to achieve constant power consumption. Cells and wires must mainly be balanced in a capacitive (identical capacitances of complementary wires and nodes) and resistive (identical charge/discharge paths) manner. Clearly, designers can never perfectly achieve this balancing, because of process variations, complex cross-coupling effects, and area limitations. Having

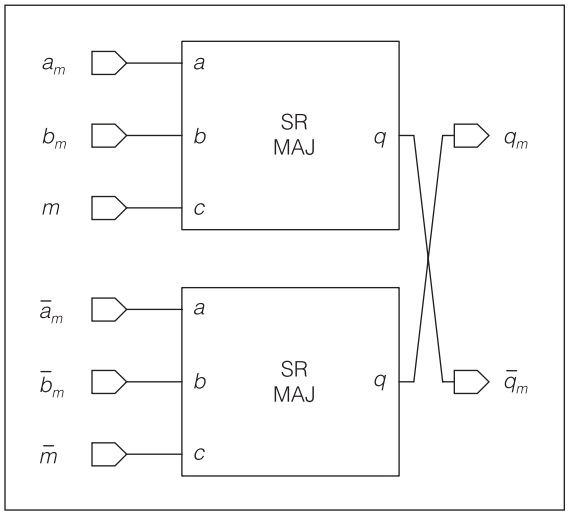

Figure 3. Schematic of the DPA-resistant masked dual-rail precharge logic (MDPL) NAND cell. This logic style uses hiding to be resistant to power analysis attacks; that is, only masked (randomized) data is processed. MDPL NAND cells are built with single-rail (SR) majority (MAJ) cells, which are available in typical standard cell libraries. (Source: Mangard, Oswald, and Popp,[2] with kind permission of Springer Science and Business Media.)

logic styles without balancing constraints would be a significant improvement. In theory, masked logic styles can solve this problem, but these styles have other limiting factors.

Second, the *memory effect* can reduce the DPA resistance of both hiding and masked logic styles. Depending on the exact cell design, internal nodes might stay in a state that depends on previous input values, thus causing data-dependent power consumption. To avoid this effect, designers must ensure that all internal nodes of a cell are precharged or masked correctly. Achieving such a functionality often requires custom cell design, which involves considerably more design effort than using available standard cells.

Obviously, if *glitches* appear in hiding logic styles, designers cannot achieve constant power consumption. Fortunately, glitches do not occur in typical hiding logic styles. For masked logic styles, it was unclear at first if glitches were a problem, but various research groups have showed that glitches cause DPA leakage in masked circuits.[8] Thus, glitches must be avoided in such circuits as well.

Fourth, researchers have recently shown that *early propagation* is a major threat to the DPA resistance of masked logic styles—specifically, for FPGAs and within the Scard project for ASICs.[9] Early propagation means that a logic cell switches its output to a new value as soon as it is determined by some of the new input values and not only after all inputs have reached their new values. The data dependency of the TOE of masked cells must be avoided in DPA-resistant circuits. Note that early propagation is also a threat to hiding logic styles, which is more obvious.

Another threat to masked circuits is the *detection of the mask value*, which lets attackers completely cancel out the effect of masking in a DPA attack. In particular, such an attack is dangerous for single-masked circuits, where only one mask value is used for all signals in the circuit. Attacks that try to determine the mask value aim at either the mask net's power consumption or the probability density function of the power consumption for different mask values.[10] To avoid such attacks, the mask net's power consumption must be constant. Furthermore, increasing the number of mask values per circuit is an option.

All in all, several publications in the past few years have shown the limits and critical design criteria for power-analysis-resistant logic styles. To develop secure devices, it is crucial for designers of cryptographic devices to be aware of these issues.

ALTHOUGH RESEARCHERS HAVE been working on power analysis attacks for almost a decade, so far they have not found an ultimate solution to protect cryptographic devices. Each countermeasure has its pros and cons. The countermeasures we have discussed here provide a reasonable level of protection against attacks. However, designers must be aware that there are limitations and many remaining design challenges when implementing these countermeasures. In practice, cryptographic devices, therefore, typically include several countermeasures to provide a high level of protection against power analysis attacks. ■

References

1. P.C. Kocher, J. Jaffe, and B. Jun, "Differential Power Analysis," *Proc. 19th Ann. Int'l Cryptology Conf. Advances in Cryptology*: (CRYPTO 99), LNCS 1666, Springer-Verlag, 1999, pp. 388-397.
2. S. Mangard, E. Oswald, and T. Popp, *Power Analysis Attacks: Revealing the Secrets of Smart Cards*, Springer, 2007.
3. S. Chari, J.R. Rao, and P. Rohatgi, "Template Attacks," *Proc. 4th Int'l Workshop Cryptographic Hardware and Embedded Systems* (CHES 02), LNCS 2523, Springer, 2003, pp. 13-28.
4. "FIPS-197: Advanced Encryption Standard," Nat'l Inst. of Standards and Technology, Nov. 2001; http://www.itl.nist.gov/fipspubs.
5. J.J.A. Fournier et al., "Security Evaluation of Asynchronous Circuits," *Proc. 5th Int'l Workshop Cryptographic Hardware and Embedded Systems* (CHES 03), LNCS 2779, Springer, 2003, pp. 137-151.
6. K. Tiri, M. Akmal, and I. Verbauwhede, "A Dynamic and Differential CMOS Logic with Signal Independent Power Consumption to Withstand Differential Power Analysis on Smart Cards," *Proc. 28th European Solid-State Circuits Conf.* (ESSCIRC 02), IEEE Press, 2002, pp. 403-406.
7. T. Popp and S. Mangard, "Masked Dual-Rail Pre-Charge Logic: DPA-Resistance without Routing Constraints," *Proc. 7th Int'l Workshop Cryptographic Hardware and Embedded Systems* (CHES 05), LNCS 3659, Springer, 2005, pp. 172-186.
8. S. Mangard, T. Popp, and B.M. Gammel, "Side-Channel Leakage of Masked CMOS Gates," *Proc. Topics in Cryptology: Cryptographers' Track at RSA Conf.* (CT-RSA 05), LNCS 3376, Springer, 2005, pp. 351-365.
9. D. Suzuki and M. Saeki, "Security Evaluation of DPA Countermeasures Using Dual-Rail Pre-charge Logic Style," *Proc. 8th Int'l Workshop Cryptographic Hardware and Embedded Systems* (CHES 06), LNCS 4249, Springer, 2006, pp. 255-269.

10. K. Tiri and P. Schaumont, "Changing the Odds against Masked Logic," *Proc. 13th Int'l Workshop Selected Areas in Cryptography* (SAC 2006), LNCS 4356, Springer, 2007, http://rijndael.ece.vt.edu/schaum/papers/2006sac.pdf.

Thomas Popp is a researcher at the Institute for Applied Information Processing and Communications (IAIK) and a PhD student at Graz University of Technology. His research interests include power analysis attacks on smart cards and corresponding countermeasures at the cell level. Popp has a Dipl-Ing in computer engineering from Graz University of Technology.

Stefan Mangard is a security specialist in the Security Innovation Group at Infineon Technologies Munich. His research interests include cryptography and all kinds of implementation attacks, including probing, fault induction, power analysis, and timing attacks. Mangard has a PhD in computer engineering from Graz University of Technology.

Elisabeth Oswald is a lecturer in the computer science department at the University of Bristol. Her research interests mainly focus on power analysis attacks and countermeasures, but she is also interested in other areas of cryptography. Oswald has a PhD in mathematics from Graz University of Technology.

■ Direct questions and comments about this article to Thomas Popp, Inst. for Applied Information Processing and Communications (IAIK), Graz Univ. of Technology, Inffeldgasse 16a, 8010 Graz, Austria; thomas.popp@iaik.tugraz.at.