

DPA Leakage Evaluation and Countermeasure Plug-in at logic level

Tang Ming^{1,2}, Wang Xiaorong², Qiu Zhenlong², Gao Si², Zhang Huanguo^{1,2}, Wu
Qianhong^{1,2}

¹(State Key Lab. of AIS&TC, Ministry of Education, Wuhan University in China, m.tang@126.com)

²(School of Computer, Wuhan University, Wuhan 430072, China)

Abstract:

There exist 3 different types of research about SCAs, such as SCA analysis, SCA evaluation and SCA countermeasures. All of these studies try to establish more security in cryptographic software, hardware and system. Evaluation of SCA tries to find factors of different SCAs, moreover, the purpose of SCA Evaluation could be regarded as the first step of building countermeasures against SCAs. We choose DPA, which is one of the most popular and realistic SCAs at present, as our research target to build practical evaluation scheme and countermeasure which can be regarded as plug-in of EDA toolkits and could help designers of circuits to judge the power leakage and improve the resistance against DPAs automatically. Our contribution concludes: more accurate evaluation scheme; more efficient balanced scheme; be portable to build countermeasures based on evaluation scheme, furthermore, our countermeasures could be plug in EDA toolkits which is automatic and transparent to designers of circuits.

Keywords: DPA, DPA evaluation, DPA countermeasure, Karnaugh Diagram

1. Background

SCA evaluations can be classified into three categories: algorithm-level, logic-level, and gate-level. There could be other different classifications such as circuit-level and logic-level, and algorithm-level and circuit-level. The logic-level

corresponds to logic function implementation. The circuit-level considers the position and route of a real circuit.

For a block cipher, an algorithm-level evaluation mainly targets on non-linear operations. Prouff considered non-linear operations as the core of DPA and introduced the concept: “transparency order of an S-box” to evaluate the resistance against DPAs . However, the concept of transparency order cannot be used for all types of circuits except pre-charge circuits.

Suzuki, Saeki, and Ichikawa provided an elegant logic-level evaluation explored the qualitative evaluation of WDDL, MASK-AND, etc [22]. Circuit-level evaluations and countermeasures such as aim to eliminate the leakage in the phase of place and route. These studies judge the leakage from different levels and phases in an implementation of cryptographic algorithms.

From target of countermeasures against DPA, there are two different types of schemes which are used in power collection [20] and power analysis [24,25]. The countermeasures in power collection try to eliminate the leakage power information, however, the schemes in power analysis are aim to increase complexity of analyzing for DPA attackers. Moreover, countermeasures in power analysis are more dependent on special DPA, for they rely on the concrete DPA analysis model, which means different countermeasure in power analysis corresponds to different DPA analysis.

Naturally, countermeasures in power collection are more ideal to design security circuits, for they could protect circuits from DPAs completely and could resist any types of power attacks such as DPAs, CPA, or other HODPAs.

Unfortunately, existing countermeasures in power collection such as WDDL, dual rail are not able to resist power attacks successfully. Moreover, there are no countermeasures combined with EDA design tools to help designers to acquire safety circuits immune to SCAs.

The work of Regazzoni, Cevrero, Standaert considers a real circuit design flow and uses the conditional entropy between the secret key and its corresponding leakage [17]. The evaluation flow is based on the simulation. Their countermeasure requires adding noise to different parts of circuits. The experimental results are based on the

evaluation of different noise injections.

The purpose of this paper is to compare different protected designs from area and power consumed.

The contribution of this paper is the first one to “develop a design flow based on standard CAD tools that can automatically synthesize and place-and-route such hybrid designs.” However, the research of this paper stops in evaluation step, not to extend this scheme to automatically design safety circuits against DPAs.

The other parts of this paper conclude: section 2 introduces our evaluation of logic circuit and the contribution of this evaluation to our Karnaugh Diagram countermeasure; section 3 describes our Karnaugh Diagram method carefully and illustrate the implementation of our countermeasure; we design and make extensive experiments to test the efficiency and effectiveness of our Karnaugh Diagram method in section 4; section 5 gives the final conclusion of this paper.

2. Evaluation

2.1 leakage model

In this paper, leakage model means evaluation of the number of leakage information. Moreover, a leakage model is usually proposed to resist the special type of SCA, such as the Hamming weight of the secret value corresponding to dual-rail coding countermeasure [20].

However, Hamming weight has been proofed to hardly cover most types of circuits except pre-charge circuits. For Hamming weight of a string is the number of symbols that are different from the zero-symbol of the alphabet used, which corresponds to the property of pre-charge circuits whose original is constant and often be set to zero.

In order to evaluate the power leakage of a circuit, more general leakage model: the switch model has been proposed and used in many EDA tools, such as Primepower in Synopsys [1]. Switch model is based on the power of a circuit is linear with the number of transverse gates, which include two different switch situations: 0->1 and 1->0. Our evaluation and countermeasure are both based on switch model.

2.2 Our evaluation

Point: the contribution and relation of our evaluation to our countermeasure

1.2.1 Introduction of our evaluation

Considering the property of Karnaugh Map, we make some modifications to improve the efficiency of evaluation.

We make the final evaluation about DPA in logic circuit level as following:

When the *st_kano* has the same number of “one ” units and “zero” units, the logic circuit has no DPA leakage, else, it has DPA leakage. This evaluation is much more efficient than true table evaluation. Furthermore, we prove the effectiveness of this evaluation in theory 4.

Theory 4: When the *st_kano* has the same number of “one ” units and “zero” units, the logic circuit has no DPA leakage, else, it has DPA leakage.

Proof of Theory 4:

- 1) firstly to certify the correctness of theory 4 in 2-input logic function

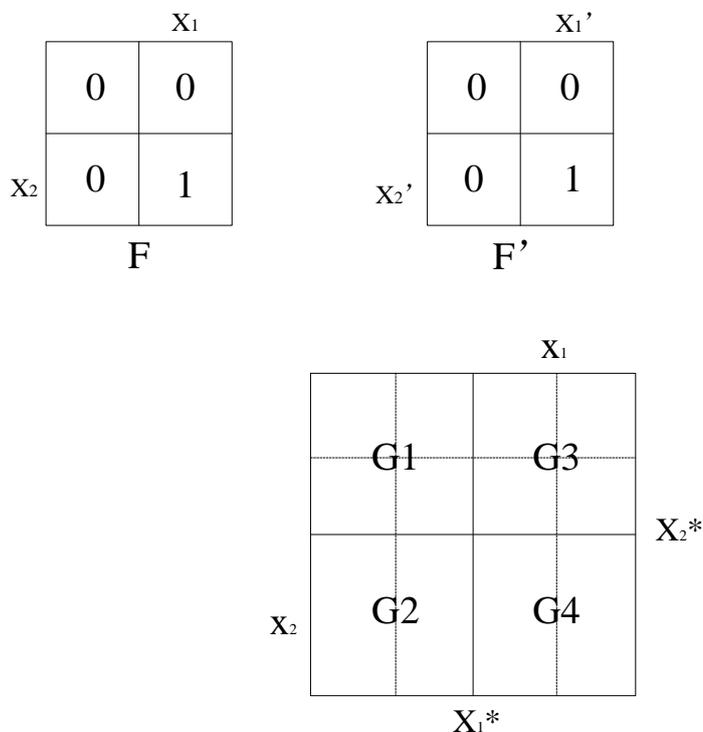


Figure 4: evaluation based on *st_kano*

P1. *sw_kano* table is cut into 4 parts { G1, G2, G3, G4} based on X_1

partition as in figure 4.

P2. based on the establish guidelines of sw_kano, we can get:

$$\begin{aligned}G1 &= l1r1 \wedge t2 \\G2 &= l1r2 \wedge t2 \\G3 &= l2r1 \wedge t2 \\G4 &= l2r2 \wedge t2\end{aligned}\tag{1}$$

Where $lirj$ represents one unit in st_kano whose column index is i and row index is j ; $t2$ is the right st_kano in figure 4.

P3. There are two relationships between $l1r1$ and $l1r2$

First: $l1r1 = /l1r2$

Because $t2$ has the same number of “one” units and “zero” units, we can get: $G1+G2=G3+G4$, which means the number of “one” units in $G1$ and $G2$ equals to that in $G3$ and $G4$.

Second: $l1r1 = l1r2$

Then $l2r1 = l2r2$, and $l1r1 = /l2r1$, we can get:

$$G1 = G2 \text{ and } G3 = G4$$

Considering the establishment of G_i in function (1), we can get the number of “one” units in $G1$ equals to the number of “zero” units in $G3$. with the same knowledge, the number of “one” units in $G2$ equals to the number of “zero” units in $G4$.

Supposing that $l1r1=0$, $l2r1=1$, therefore, the number of “one” units in $G1$ equals the number of “one” units in $t2$, and the number of “zero” units in $G3$ equals to the number of “zero” units in $t2$.

Because $t2$ has the same number of “one” units and “zero” units, we can get: $G1+G2=G3+G4$, which represents $X1$ is balanced in different partitions.

This process can be expanded to other original input bits. Theory 4 is true in 2-input logic functions.

2) finally to proof the theory 4 in any number of input logic function

P1. We choose $\{X1, \dots, Xn\}$ as n input bits of a logic function;

P2. We define $\{Gi1, Gi2, Gi3, Gi4\}$ as the groups of units which corresponds to $X1$ different partitions with Xi in sw_kano .

P3. Based on the conclusion of 2-input logic function previously, we can get

$$Gi1+Gi2=Gi3+Gi4$$

P4. Therefore, “one” partition of $X1$ in sw_kano can be represented as

$$\sum_{i=2}^n (Gi1 + Gi2) , \text{ while “zero” partition of } X1 \text{ in } sw_kano \text{ is}$$

$$\sum_{i=2}^n (Gi3 + Gi4) .$$

P5. As the results of P3, we can get $\sum_{i=2}^n (Gi1 + Gi2) = \sum_{i=2}^n (Gi3 + Gi4) .$

Therefore, as for a logic function theory 4 is true. We can use theory 4 to evaluate DPA leakage efficiently.

2.2.3 The relationship between our evaluation and countermeasure

The core of our evaluation is to count the number of switched gates in whole circuits. This scheme is built in true table of logic function of the circuit designing, which could be regarded as the cipher algorithm. Moreover, our countermeasure which will be illustrated in Section 3 is based on a Karnauth Map to correspond to the logic circuit implementation. As we all know, true table and Karnauth Map are one-to-one relationship, therefore, it is very convenient to transform from logic function in algorithm to logic circuit in implementation during building countermeasure of DPAs. Furthermore, it is very easy combine both true table and Karnauth Map with EDA tools of the circuit design flow.

3. Our Countermeasure

3.1 Karnaugh Map Countermeasure

As many resistances against DPAs, the purpose of our countermeasure is to hold the leakage of power constant. Based on our evaluation of true table, we can find every output bit including intermediate output bits which have the possibilities of

power leakage. Considering the direct relationship between Karnaugh Map and logic function, and original input bit partition of DPA could be very convenient decrypted on Karnaugh Map. We decide to introduce Karnaugh Map into our countermeasure against DPAs.

In order to introduce our countermeasure explicitly, the symbols are introduced as following:

st_kano: standard Karnaugh Map corresponds to the logic function of original circuits;

sw_kano: it is the combination of two st_kano, and it is a changed Karnaugh Map which is used to judge balance level of circuits;

Bu_stsw: guidelines of establish sw_kano from two st_kano;

Po_bala: guidelines of balancing operation on sw_kano;

Our countermeasure consists of 3 procedures which include Karnaugh Map establishment, sw_kano balance, and st_kano balance.

3.2 Karnaugh Map Establishment

1) Evaluation based on true table

The details of true table have been introduced in section 2.

As the results of true table, the output bits have the power leakage and their corresponding logic functions which are also the output of synthesis netlist are made flag.

2) Build standard Karnaugh Map

- Karnaugh Map: is a method to simplify Boolean algebra expressions. The Karnaugh map reduces the need for extensive calculations by taking advantage of humans' pattern-recognition capability, permitting the rapid identification and elimination of potential race conditions.
- Build st_kano: as the definition of st_kano introduced, st_kano corresponds to the logic function of the original circuits.

There are two different schemes to build st_kano:

One is based on the definition of standard Karnaugh Map. Set each unit in

st_kano to “1” which is corresponding to the existing element in logic function. The first scheme is same as the traditional method of standard Karnaugh Map generation.

Actually, the logic description in st_kano is different from original logic function, except that they have the same logic values in output bits. Because each unit in st_kano or in Karnaugh Map means the minimum element of logic function, however, an element in original logic function may be a simplify logic element which could correspond a few minimum logic elements. If we concern the number of gates, the description of st_kano is different from that in original logic function.

Therefore, we design another scheme to establish st_kano. We use true table to establish st_kano directly. After the operation of true table, we can find every output bit which has leakage possibility. We set the corresponding units of these output bits in st_kano, for true table holds the relationship between original input and all output bits. We give an example as following, table 1 is the true table of this example and figure 1 is the st_kano:

Table 1: build st_kano based on true table

DPA leakage

X1 or X2 partition

AND2

X1	X2	X1'	X2'	$\Delta X1$	$\Delta X2$	F	F'	ΔF	
0	0	0	0	0	0	0	0	0	
0	0	0	1	0	1	0	0	0	
0	0	1	0	1	0	0	0	0	
0	0	1	1	1	1	0	1	1	$P(\Delta F=1 X1=0)=2/8=0.25$
0	1	0	0	0	1	0	0	0	
0	1	0	1	0	0	0	0	0	
0	1	1	0	1	1	0	0	0	

0	1	1	1	1	0	0	1	1
1	0	0	0	1	0	0	0	0
1	0	0	1	1	1	0	0	0
1	0	1	0	0	0	0	0	0
1	0	1	1	0	1	0	1	1
1	1	0	0	1	1	1	0	1
1	1	0	1	1	0	1	0	1
1	1	1	0	0	1	1	0	1
1	1	1	1	0	0	1	1	0

$P(\Delta F=1|X_1=1)=4/8=0.5$

Table 1 is the true table of $F = X_1 \bullet X_2$, where $F' = X_1' \bullet X_2'$, $\Delta X_1 = X_1 \oplus X_1'$, $\Delta X_2 = X_2 \oplus X_2'$, and output bit F has power leakage for $P(\Delta F=1|X_1=0) \neq P(\Delta F=1|X_1=1)$. Then we should build st_kano of this output bit F as in figure 1.

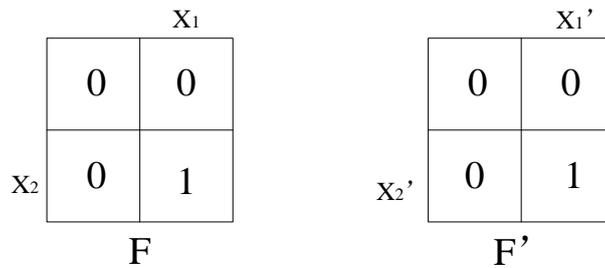


Figure 1: st_kano establishment

Figure 1 is the two st_kanos of logic function F, where F and F' correspond to neighboring state of this logic function respectively. F is the previous state and F' is the second one. The reason for two kanos is because we choose switch model as power leakage model of DPA evaluation which need count the number of switched gates. As we all known, switching gate depends on the different states such as 1 to 0 or 0 to 1, therefore, two st_kanos represent these two neighboring states.

3) Establish switch Karnaugh Map

For st_kano is another illustration of original logic functions, we use sw_kano which is the switch Karnaugh Map to represent the switched gates of circuits in logic level. Moreover, sw_kano is based on st_kanos. Because two neighboring st_kanos could induce the present and next state of each gates. Figure 2 is an example of the sw_kano which is corresponding to the logic function of Figure 1.

		X_1		
		0	0	
		0	1	
		0	0	
		0	1	
		1	1	
		1	0	
				X_2^*
X_2				
				X_1^*

Figure 2: sw_kano corresponding to st_kano in Figure 1

From the instance in Figure 2, we conclude the guidelines of sw_kano establishment as following:

- Every related units in two neighboring st_kano are combined to one unit of sw_kano. The related units are the two units in two st_kanos to generate one unit of sw_kano. We choose “row first” order to search each unit in st_kano, which means the index order of row is from small to big and the index order of column is also small to big in one row.
- Supposing that the size of st_kano is $m*n$, sw_kano built on two neighboring st_kanos has the size of m^2*n^2 .
- The units of sw_kano only have two different values which are one and zero. If the related units of two neighboring st_kanos have different values, the corresponding unit in sw_kano is one, if the related units are same, it will generate zero unit in sw_kano.
- Addresses of units in sw_kano are also based on the related units of st_kanos. As to the address of one unit in sw_kano equals to the combination of two addresses of related units in st_kanos. For example, if the addresses of related

units are x_1x_2 , $x_1^*x_2^*$, the address of corresponding unit in `sw_kano` is x_1x_2 $x_1^*x_2^*$.

- Although original Karnaugh Map has two dimensions: row direction and column direction, we design both `st_kano` and `sw_kano` in one dimension array to reduce address space and increase efficiency of the resistance scheme.

3.3 Balance Guidelines of switch Karnaugh Map

This phase consists of two steps: the first is re-evaluation of `sw_kano`, for the logic circuits have different gates in Karnaugh Map which is illustrate in second step of our countermeasure scheme; and the second is balance logic design to eliminate power leakage.

Therefore, we should establish two sets of guidelines corresponding evaluation and balance respectively.

1) Evaluation guidelines of `sw_kano`

P1. Each address bit of `sw_kano` unit is the original input bit. Therefore, each address bit partition decides the extent of power leakage.

P2. To count the number of “1” value unit in `sw_kano` for each address bit at different partitions. If the counter of one partition equals to that of zero partition, it is no power leakage of this output at the corresponding original input bit. If they are not equal, we should record these original input bits to do balance operations.

For example, x_1 partition of `sw_kano` in Figure 2, the counter of $x_1=1$ partition is 4, and the counter of $x_1=0$ partition is 2. The two counter values are not equal, therefore, F function will leakage power in x_1 partition.

2) Problems in `sw_kano` balance

There are some key problems to resolve in `sw_kano` balance. We analyze these problems which are symbolized as Q and introduce our resolutions as A as following:

- Q1: When we find an output bit had power leakage in an original input partition, which means that the number of switched gates in the logic function

of this output bit is different in partition 1 with partition 0. Supposing that the number of switched gates in partition 1 is more than that in partition 0, the question is whether we should increase switched gates in partition 0, or to decrease that in partition 1?

A1: Considering that if we decrease switched gates in partition 1, it will bring another question that this handle operation may change the logic function of original design of this output bit. Therefore, we choose to increase switched gates in less number partition.

- Q2: As the same question in Q1, increase in the number of switched gates in partition 0 may change the logic function of original design.

A2: In order to delete the influence of original design, we regard the increase operation as the logic function of a new output bit. We call these new output bits in this kind of increase as balanced output bits.

- Q3: Though the balanced output bits could get rid of the influence of original design, it is still a question caused by these bits. As we all known, only logic function except the final output bits of circuit design must be optimized in synthesis phase of EDA designing.

A3: Therefore, we should create balanced output bits as final output bits of original design. Of course, we could make some combination of these balanced output bits to decrease the number of total final output bits, which will improve the hardware resource and the power consumption.

- Q4: However, these balanced output bits could not be added in original circuit simply. Because, every balanced output bit corresponds to a unbalanced output bit in original circuit. The corresponding relationship has time character, which means that each balanced output bit must be synchronous with its corresponding unbalanced output bit.

A4: Based on the true table in the previous part of this section, we could acquire the level of each unbalanced output bit. Each balanced output bit should be organized in its corresponding level. If there is a level of original circuit has no unbalanced output bits, we could add some NULL operations

without any logic circuits but take one clock period.

- Q5: The balanced measure on one original input partition may influence or break the balanced relationship of other original input bits.

A5: We should set a counter for each output bit in the same logic level. Based on the values of these counters, we establish the logic functions of balanced output bits.

In order to establish an effective and efficient balance policy of sw_kano , we make and proof the truth of Theory 1 and Lemma 1.

Some symbols should be declared at first in Table 2.

- Original input bit: original input bit of logic circuits or logic functions
- $Count_Oi_Xjp$: represents the number of unit in sw_kano with one value, which corresponds to the original input bit with different partitions, where O_i is the i th output bit; X_j means the j th original input bit; p represents “1” partition ($p=1$), or “0” partition ($p=0$).
- $Dv_Xj = |Count_Oi_Xj1 - Count_Oi_Xj0|$, which means the distance of different partitions of the j th original input bit.
- Md_v : represents the distance of the present $\max Dv_xi$ with other Dv_xj ($i=j$)

For example: a two input bits logic function as following.

$$Dv_X1=4$$

$$Dv_X2=2$$

$$Md_v=2$$

Theory 1: For a logic function with any number of input bits, any original input bit with both one and zero partition have the *even* number of units in sw_kano .

Lemma 1: For a logic function with any number of input bits, Md_v in any time is even.

Naturally, when Theory 1 is true, Lemma 1 will be true too. Because $Count_Oi_Xjp$ is even, Dv_Xj is also even.

Proof of Theory 1:

a) proof the theory 1 of two original input bits of logic function

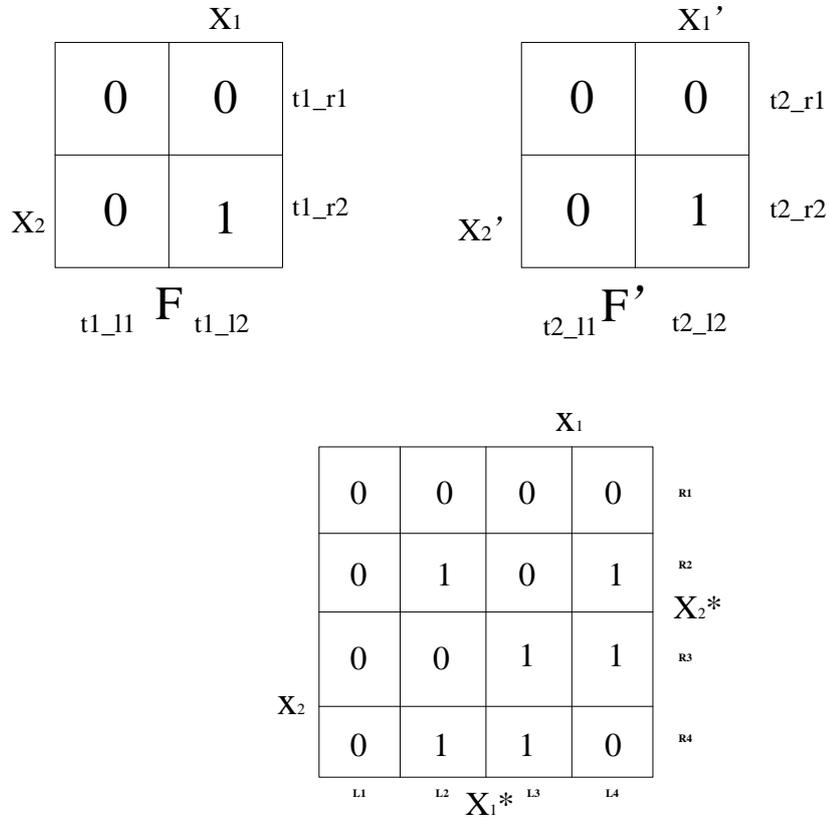


Figure 3: 2-input logic function in sw_kano

Where $t1$ represents the left st_kano, $t2$ means the right st_kano, t_{i_lj} and t_{i_rj} represent the columns and rows of $t1$ and $t2$ respectively, and L_i and R_j mean the columns and rows in sw_kano respectively.

- P1. For X_1 as an example, the one partition of X_1 in sw_kano takes L_3 and L_4 columns and the zero partition of X_1 takes L_1 and L_2 in sw_kano.
- P2. The units of L_3 and L_4 are generated by the XOR operation between $t2_{r1}$ and every unit in $t2$, and between $t2_{r2}$ and every unit in $t2$. For the number of st_kano units is 2^n for n original input bits of the logic function, the number of st_kano units is even.
- P3. There are only two relationships between $t2_{r1}$ and $t2_{r2}$.

First: $t2_{r1} = t2_{r2}$

Then the number of the units with “one” value generated by $t2_{r1}$ equals to the number by $t2_{r2}$. Therefore, the number of switched units in

L3 and L4 is even.

Second: $l2r1 = /l2r2$

Then the units with value “one” generated by $l2r1$ corresponds to the units with value zero generated by $l2r2$. Therefore, the number of switched units in L3 and L4 is 2^n and also is even.

P4. With the same situation, we can proof the other original input bits of the 2- input logic function.

b) Proof the theory 1 of any number of input bits in logic function.

P1. Supposing that the number of original input bits is n , and $X1$ is also the object input bit.

P2. The destination is to proof the “one” partition of $X1$ has even switched units in sw_kano .

P3. Supposing $lir1$ and $lir2$ be the unit of $X_i=0$ and $X_i=1$ respectively, where X_i is the any input bit of logic function except $X1$.

P4. There are only two different relationships between $lir1$ and $lir2$

First: $lir1 = lir2$

Second: $lir1 = /lir2$

With the same reason, the number of one partition of $X1$ in sw_kano related with X_i is even.

P5. Therefore, the number of one partition of $X1$ in sw_kano is even.

Theory 1 is true.

3) Balance guidelines of sw_kano

Based on these questions and their corresponding answers, we conclude balance guidelines of sw_kano as following:

- We choose to increase the less number of switched gates in the partition.
- We regard the increased units in sw_kano be the components of logic function which is contributed to the new output bits. We called these new output bits as balanced output.

- We should make a combination of every balanced output bit to decrease the number of the new final output bits.
- The combination of balanced output bits should be complied with the level of their corresponding unbalanced output bit. This kind of level could be acquired in true table establishment.
- We should forbid the influence between balanced output bits and unbalanced output bits during balanced process. The establishment of balance operations should be based on the counter for the switched gates of each output bit.

In order to illustrate sw_kano balance method clearly, we conclude the detailed the procedure of sw_kano as following.

(1) We classify the unbalanced situation in sw_kano as two types.

First: All original input bits have the same number of Dv_{xj} for their own partitions.

Second: There are different Dv_{xj} values for different original input bits.

(2) If original input bits have the same number of Dv_{xj} , we can choose the corresponding units which are satisfied with the Dv_{xj} ;

(3) For those balanced original input bits, we define these bits in the balanced output logic function with both positive and negative logic in the same times.

(4) If the number of satisfied units in sw_kano is less than Dv_{xj} , we repeat these satisfied units more than one time to receive Dv_{xj} .

(5) If the satisfied units in sw_kano are already the “one” units, balanced output logic function can also use them, for balanced output bits are independent with the original logic function.

(6) If logic function has different Dv_{xj} for different original input bits, we can repeat (2) to (5) steps from the min Dv_{xj} at present until all $Dv_{xj}=0$.

3.4 Balance Guidelines of standard Karnaugh Map

The standard Karnaugh Map and the switch Karnaugh Map correspond to the logic function of circuit and DPA leakage of different original input partitions respectively. St_kano helps us to descript logic circuits, while sw_kano establish the

relationship between circuits and power consumption. Based on the balance methods of sw_kano, we introduce balance guidelines of st_kano in this section.

We still use a 2-input logic function as an example. If we find that the unit $X_iX_jXu'Xv'$ is one of the balanced logic units in sw_kano, we will use this unit to generate the balanced units in st_kanos which include st1 and st2. The corresponding units in st_kanos are X_iX_j and $Xu'Xv'$, for example, X_iX_j in st1 and $Xu'Xv'$ in st2.

As the balanced unit in sw_kano, $X_iX_jXu'Xv'$ means that $X_iX_j = Xu'Xv'$, for example $X_iX_j=1$ and $Xu'Xv'=0$ (Actually $X_iX_j=0$ and $Xu'Xv'=1$ is equal to this condition). Because st1 equals to st2, both X_iX_j and $Xu'Xv'$ are minium units in 2-input logic function. Therefore we can define that $X_iX_j=1$ and $Xu'Xv'=0$ is the balanced units in st_kano. Furthermore, X_iX_j can not be in the same position in st1 as $Xu'Xv'$ in st2, we can combine them in a standard Karnauth Map. The final balanced logic function is $F' = X_iX_j + \overline{X_uX_v}$, while F' is independent with F .

However, it is not the end of balance measures of st_kano. There are some problems in practical implementation of balance guidelines.

Q1: Whether F' will introduce some new DPA leakage?

A1: Only X_iX_j and $XuXv$ exist in the logic function of F' . The other min units do not appear in F' , we could regard the other units as “X” value, which means the random value including both “1” and “0”. In another words, we can regard the “X” units as both “1” and “0”. Therefore, these units with “X” elements will generate the same number of switched gates and the same value of power consumption.

Q2: How to guarantee that the balanced circuits (F') will be synthesized with the corresponding unbalanced units (F)?

A2: After the analyzing and balancing for each unbalanced output bit, we can acquire every logic function for balanced output bits and their logic levels. Based on the increasment of each level, we combine the logic functions in the same level and generate the final output bit for this logic level. Of course, if there are n logic levels in original circuits, there are n final balanced output bits.

On the other hand, in order to hold the time complying, we use a matrix to

descript all final balanced output bits. Each final output bit has (n-1) NULL operation, because it is very hard to combine any two balanced output bit in different logic levels into one bit; and then each final output bit only take one clock period actually. In the following example, we will give the matrix for the example.

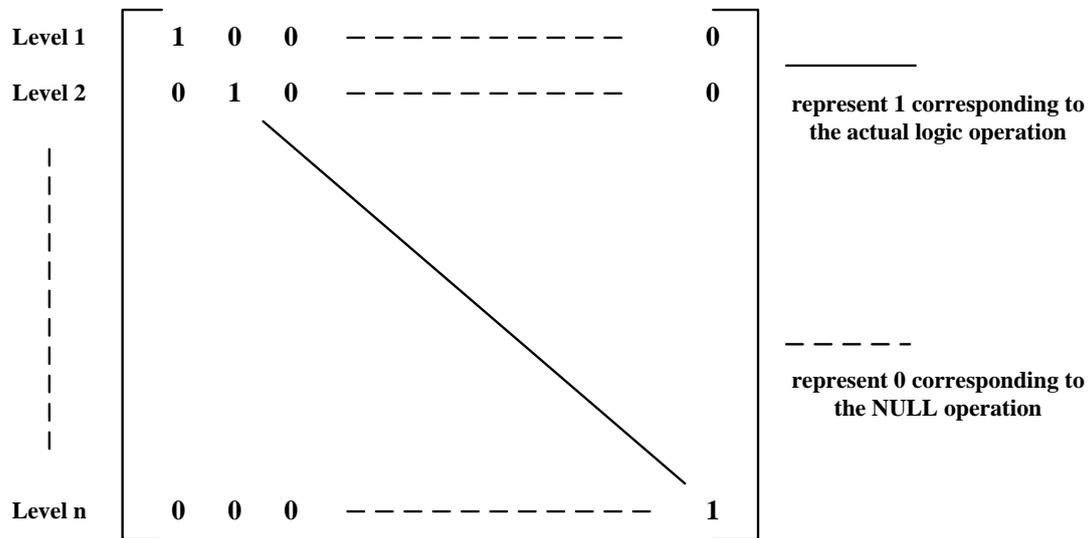


Figure 4: the matrix of final balanced output bits

Q3: If there are several balanced units in sw_kano, how to build balanced units in st_kano?

A3: When these balanced units in one sw_kano, we can confirm that they take in the same logic level. However, we can not combine these balanced units in one logic function. Because, different balanced unit corresponds to different balanced output, if we combined them into one balanced output, we will generate some other units except balanced units.

For example in a 2-input logic function, if $X_1 \bar{X}_2 \bar{X}'_1 X'_2$ and $\bar{X}_1 \bar{X}_2 \bar{X}'_1 X'_2$ are two balanced units in sw_kano, we should build two balanced functions such as $F1'$ ($F1' = X_1 \bar{X}_2 + \bar{X}'_1 X'_2$) and $F2'$ ($F2' = \bar{X}_1 \bar{X}_2 + \bar{X}'_1 X'_2$) corresponding to these two balanced units respectively. Otherwise, if we build one balanced function named as $F' = F1' + F2'$ which will generate some new switched units in the new sw_kano. This illustrates us that we can also cut one original logic function into different sub logic functions to eliminate DPA leakage. The latter method has the advantage of less power consumption than increasing balanced units.

Q4: How to compute the number of levels in balanced units? For example, balanced units are F1' and F2' in Q3. If we consider the original input as the first level of logic circuits, balanced units like F1' and F2' take more than one logic levels, which conflict with the number of logic level in one output bit.

A4: We could consider the all "min units" in logic functions as the first logic level, which can resolve the problem. We name these "min units" as "auto units" and this resolution can also fit for logic functions with different original input bits. Actually, both original input and "auto units" could be regarded as the first logic level, because the purpose of balanced guidelines for sw_kano and st_kano is to guarantee the balanced units can play role with unbalanced logic function parallel, which has been solved with the matrix in A2.

We conclude the balance guidelines of st_kano as following:

- (1) We generate the balanced units in st_kano based on the balanced units in sw_kano as previous introduction.
- (2) For the whole logic circuit, we use the matrix in figure 4 to guarantee the balanced circuits (F') will be synthesized with the corresponding unbalanced units (F).
- (3) We should not combine different balanced units in the same logic level, which will generate extra logic circuits to leakage power.

4. Experiments

4.1 conditions of experiments

General PC

CPU: Pentium (R) Dual-Core 2.7Ghz

Memory: 1024MB RAM

Operating system: Windows 7

Java Platform: jdk1.6.0_10

Java IDE: eclipse

Text Editor: UltraEdit 32

4.2 results and analyses

4.2.1、 Make the judgement about the impact of the input on transition

We simulated to determine the impact of each input on the transition. We set the number of the input while output values are random. The results are as follows:

The number of the inputs that have the impact on transition

Input number	test1	test2	test3	test4	test5	test6	test7	test8
1	0	0	0	0	0	0	0	0
2	0	2	0	0	0	2	0	0
3	3	3	3	0	1	0	1	1
4	4	0	4	0	4	2	1	0
5	5	5	4	0	1	3	1	5
6	0	4	6	4	4	4	5	6
7	7	6	7	7	7	5	6	5
8	6	8	8	7	6	7	0	8
9	9	8	8	9	8	0	9	8
10	9	10	9	10	10	8	9	10

Table: The result of the impact of the input on transition

Conclusion: Not all inputs have the influence on the transition.

4.2.2、 The level division of RTL netlist

We carry out the level division of the netlist to analyze Karnaugh map better, and get the output values for the following experiments in the meantime. Due to the space constraints, the specific experimental data is omitted here.

The experiment consists of two parts: Level division of the Present-Sbox, as well as the level division of AES.

4.2.3、 Using actual netlist file to analyze the impact of the input on transition

Using the method mentioned in 4.2.1 to analyze the generated gate-level netlist file. The output values of the gates in circuit placed in out.txt (equivalent to the above

random value). The results are shown in following Figure:

1) 、 The analysis of Present-Sbox netlist (input numbers are 4):

The table for the analysis of Present-Sbox netlist

Name	The number of the inputs that have the impact on transition
gate[0]	0
gate[1]	0
gate[2]	4
gate[3]	4
gate[4]	2
gate[5]	4
gate[6]	2
gate[7]	2
gate[8]	4
gate[9]	0
gate[10]	0
gate[11]	2
gate[12]	2
gate[13]	3
gate[14]	3
gate[15]	3
gate[16]	3
gate[17]	3
gate[18]	3
gate[19]	3
gate[20]	3
gate[21]	3
gate[22]	3

gate[23]	3
gate[24]	3
gate[25]	3
gate[26]	3
gate[27]	3
gate[28]	3
gate[29]	3
gate[30]	3
gate[31]	3
gate[32]	3
gate[33]	3
gate[34]	3
gate[35]	3
gate[36]	3
gate[37]	3
gate[38]	3
gate[39]	3
gate[40]	3
gate[41]	3
gate[42]	3

Type of logic gate	The Number of logic gate in original circuit	The Number of logic gate in WDDL	The Number of logic gate in KANO	WDDL - KANO	Contrast(better one)
AND	9	18	14	4	KANO
OR	4	8	15	-7	WDDL
NOR	11	22	15	7	KANO
NAND	5	10	14	-4	WDDL
XOR	1	2	31	-29	WDDL

NXOR	1	2	1	1	KANO
INV	8	78	8	70	KANO
DDF	4	8	5	3	KANO
Amounts	43	148	103	45	KANO

2)、 The analysis of the ASE netlist (input numbers are 8)

According to the test data, we found that basically every logic door has the inputs that influence the transition.

4.2.4、 Contrast between KANO and WDDL

We achieved to convert the original netlist into the WDDL netlist and KANO netlist, and made the comparison of the amount of components. The results are shown in Figure:

1) 、 Contrast in Present-Sbox netlist

2) 、 Contrast in AES netlist

Type of logic gate	The Number of logic gate in original circuit	The Number of logic gate in WDDL	The Number of logic gate in KANO	WDDL - KANO	Contrast(better one)
AND	249	498	374	124	KANO
OR	106	212	356	-144	WDDL
NOR	250	500	356	144	KANO
NAND	125	250	374	-124	WDDL
XOR	0	0	731	-731	WDDL
NXOR	0	0	0	0	KANO=WDDL
INV	95	1650	95	1555	KANO
DDF	8	16	9	7	KANO
Amounts	833	3126	2295	831	KANO

3)、 WDDL netlist

```

module Sbox8 ( in, clk, out );
input [7:0] in;
output [7:0] out;
input clk;
wire N293, N546, N783, N1019, N1271, N1486, N1751, N1963, n406, n407, n408,
n409, n410, n411, n412, n413, n414, n415, n416, n417, n418, n419,
n420, n421, n422, n423, n424, n425, n426, n427, n428, n429, n430,
n431, n432, n433, n434, n435, n436, n437, n438, n439, n440, n441,
n442, n443, n444, n445, n446, n447, n448, n449, n450, n451, n452,
n453, n454, n455, n456, n457, n458, n459, n460, n461, n462, n463,
n464, n465, n466, n467, n468, n469, n470, n471, n472, n473, n474,
n475, n476, n477, n478, n479, n480, n481, n482, n483, n484, n485,
n486, n487, n488, n489, n490, n491, n492, n493, n494, n495, n496,
n497, n498, n499, n500, n501, n502, n503, n504, n505, n506, n507,
n508, n509, n510, n511, n512, n513, n514, n515, n516, n517, n518,
n519, n520, n521, n522, n523, n524, n525, n526, n527, n528, n529,
n530, n531, n532, n533, n534, n535, n536, n537, n538, n539, n540,
n541, n542, n543, n544, n545, n546, n547, n548, n549, n550, n551,
n552, n553, n554, n555, n556, n557, n558, n559, n560, n561, n562,
n563, n564, n565, n566, n567, n568, n569, n570, n571, n572, n573,
n574, n575, n576, n577, n578, n579, n580, n581, n582, n583, n584,
n585, n586, n587, n588, n589, n590, n591, n592, n593, n594, n595,
n596, n597, n598, n599, n600, n601, n602, n603, n604, n605, n606,
n607, n608, n609, n610, n611, n612, n613, n614, n615, n616, n617,
n618, n619, n620, n621, n622, n623, n624, n625, n626, n627, n628,
n629, n630, n631, n632, n633, n634, n635, n636, n637, n638, n639,
n640, n641, n642, n643, n644, n645, n646, n647, n648, n649, n650,
n651, n652, n653, n654, n655, n656, n657, n658, n659, n660, n661,
n662, n663, n664, n665, n666, n667, n668, n669, n670, n671, n672,
n673, n674, n675, n676, n677, n678, n679, n680, n681, n682, n683,

```

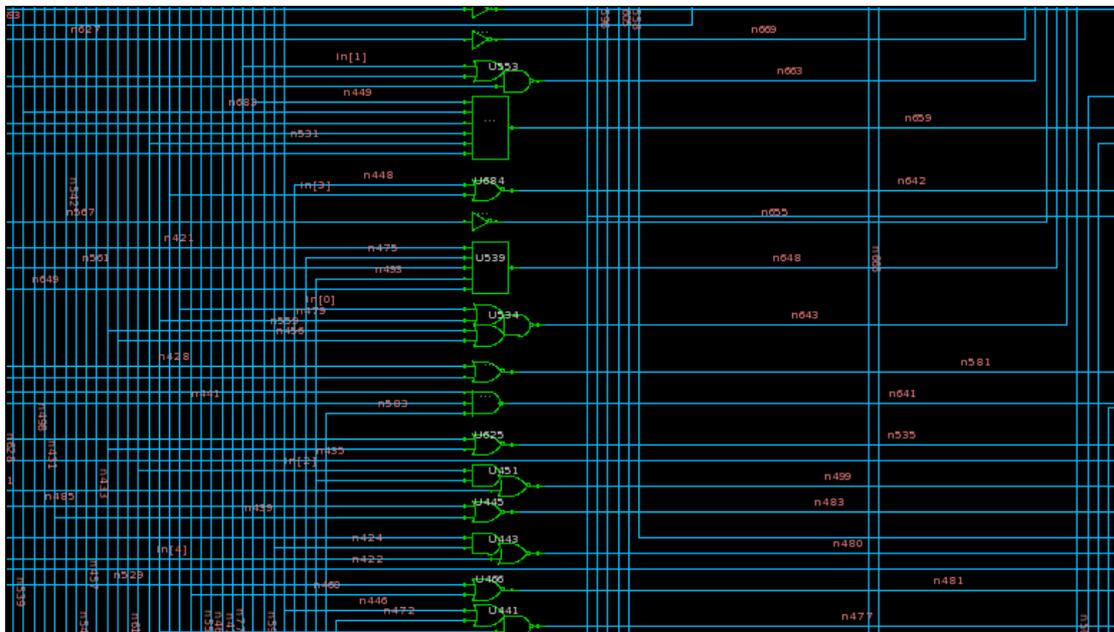
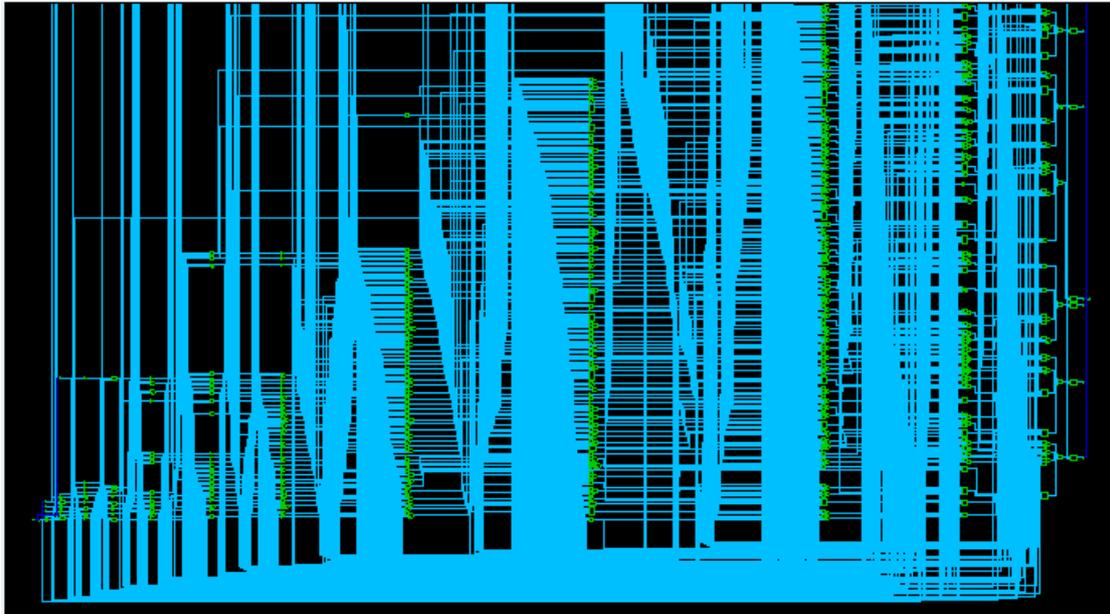
4)、 KANO netlist

```

module Sbox8 ( in, clk, out);
input [7:0] in;
input clk;
output [7:0] out;
output Nout;
wire N293, N546, N783, N1019, N1271, N1486, N1751, N1963, n406, n407, n408, n409, n410, n411, n412, n413, n414, n415, n416
temp3, U514temp1, U514temp2, U515temp1, U515temp2, U515temp3, U517temp1, U517temp2, U518temp1, U519temp1, U519temp2, U520t
, NNtemp68, Ntemp69, NNtemp69, Ntemp70, NNtemp70, Ntemp71, NNtemp71, Ntemp72, NNtemp72, Ntemp73, NNtemp73, Ntemp74, NNtemp
NNtemp266, Ntemp267, NNtemp267, Ntemp268, NNtemp268, Ntemp269, NNtemp269, Ntemp270, NNtemp270, Ntemp271, NNtemp271, Ntemp
NNtemp461, Ntemp462, NNtemp462, Ntemp463, NNtemp463, Ntemp464, NNtemp464, Ntemp465, NNtemp465, Ntemp466, NNtemp466, Ntemp4
Ntemp656, Ntemp657, NNtemp657, Ntemp658, NNtemp658, Ntemp659, NNtemp659, Ntemp660, NNtemp660, Ntemp661, NNtemp661, Ntemp66
IN VX1 U416 ( .A(in[0]), .Y(U416temp0) );
IN VX1 U422 ( .A(in[0]), .Y(n428) );
IN VX1 U426tempg1 ( .A(in[0]), .Y(U426temp1) );
IN VX1 U432tempg1 ( .A(in[0]), .Y(U432temp1) );
AND2X1 U437tempg0 ( .A(in[3]), .B(in[4]), .Y(U437temp1) );
IN VX1 U446tempg1 ( .A(in[0]), .Y(U446temp1) );
IN VX1 U450 ( .A(in[0]), .Y(U450temp0) );
IN VX1 U457tempg1 ( .A(in[0]), .Y(U457temp1) );
IN VX1 U460tempg1 ( .A(in[0]), .Y(U460temp1) );
IN VX1 U464tempg1 ( .A(in[0]), .Y(U464temp1) );
IN VX1 U466 ( .A(in[0]), .Y(n530) );
IN VX1 U471 ( .A(in[0]), .Y(U471temp0) );
IN VX1 U473tempg1 ( .A(in[0]), .Y(U473temp1) );
IN VX1 U487 ( .A(in[0]), .Y(U487temp0) );
AND2X1 U488tempg0 ( .A(in[1]), .B(in[2]), .Y(U488temp1) );
OR2X1 U495tempg0 ( .A(in[3]), .B(in[1]), .Y(U495temp1) );
IN VX1 U497tempg1 ( .A(in[0]), .Y(U497temp1) );
IN VX1 U502 ( .A(in[0]), .Y(n469) );
OR2X1 U508tempg0 ( .A(in[7]), .B(in[6]), .Y(U508temp1) );
IN VX1 U512 ( .A(in[0]), .Y(n613) );

```

5) circuit diagram



Conclusion: KANO has obvious advantages in the circuit resources, and of course the circuit power consumption will be less than WDDL, so KANO achieves to reduce the circuit resources and power consumption.

We found the phenomenon that some logic gates themselves are in balance and thus we carried out a method against DPA based on the Karnaugh map --KANO. Contrast to WDDL, KANO largely reduces the amount of circuit resources and the power consumption of transition. KANO is a more lightweight approach against DPA, which can be better applied to practice.

5. Conclusion

We propose a new evaluation and countermeasure for Differential Power Attack. Based on Karnaugh Diagram, we design more accurate evaluation and more lightweight countermeasure of DPA. The core of these interesting contributions relies on our finding of the relationship between Kano and logic circuits. We improve this kind of relationship, and create a new kind of Karnaugh Diagram to illustrate the power leakage and even the differential power value leakage. The latter is the factor of making our countermeasure more efficiency. We have proofed the effectiveness of our evaluation and countermeasure, evaluated and compared the efficiency of this new countermeasure with WDDLs.

Reference:

- [1] http://www.synopsys.com/support/li/installation/documents/install072/primepower-y-2006.06_install.pdf/.
- [2] D. Agrawal, B. Archambeault, J. R. Rao, and P. Rohatgi. The EM Side-Channels(s): Attacks and Assessment Methodologies. In CHES 2002. LNCS 2523, 2003.
- [3] R. M. Avanzi. Countermeasures against Differential Power Analysis for Hyperelliptic Curve Cryptosystems. In CHES 2003, pages 366-381. LNCS 2779, 2004.
- [4] E. Biham and A. Shamir. Differential Fault Analysis of Secret Key Cryptosystems. In Proceedings of Crypto'97, pages 513-525. LNCS 1294, 1998.
- [5] E. Brier, C. Clavier, and F. Olivier. Correlation Power Analysis with a Leakage Model. In CHES 2004, pages 16-29. LNCS 3156, 2004.
- [6] M. Bucci, L. Giancane, R. Luzzi, and A. Triletti. Three-Phase Dual-Rail Pre-charge Logic. In CHES 2006, pages 232-241. LNCS 4249, 2006.
- [7] S. Dziembowski and K. Pietrzak. Leakage-Resilient Cryptography in the Standard Model. <http://eprint.iacr.org/2008/240.pdf>.
- [8] C. H. Kim and J.-J. Quisquater. New Differential Fault Analysis on AES Key Schedule: Two Faults Are Enough. In CARDIS 2008, pages 48-62. LNCS 5189, 2008.

- [9] P. Kocher. Timings Attacks on Implementations of Di_e Hellman, RSA, DSS and other systems. In *Crypto 1996*, pages 104-113. Springer-Verlag, 1996.
- [10] P. Kocher, J. Ja_e, and B. Jun. Di_ifferential Power Analysis. In *Crypto 1999*, pages 338-397. LNCS 1666, 1999.
- [11] L. Lin, M. Kasper, T. G_ uneyesu, C. Paar, and W. Burleson. Trojan Side-Channels: Lightweight Hardware Trojans through Side-Channel Engineering. In *CHES 2009*, pages 382-395. LNCS 5747, 2009.
- [12] F. Menichelli, R. Menicocci, M. Olivieri, and A. Tri_letti. High-Level Side-Channel Attack Modeling and Simulation for Security-Critical Systems on Chips. In *IEEE Transactions on Dependable and Secure Computing 2008*, pages 164-176. IEEE 5(3), 2008.
- [13] T. S. Messerges. Using Second Order Power Analysis to Attack DPA Resistant Software. In *CHES 2000*, pages 238-251. LNCS 1965, 2000.
- [14] T. S. Messerges, E. A. Dabbish, and R. H. Sloan. Investigations of Power Analysis Attacks on Smart Cards. In *CHES 1999*. LNCS 1666, 1999.
- [15] E. Peeters, F.-X. Standaert, N. Donckers, and J.-J. Quisquater. Improved Higher-Order Side-Channel Attacks with FPGA Experiments. In *CHES 2005*, pages 309-323. LNCS 3659, 2005.
- [16] E. Prouff. Dpa Attacks and S-Boxes. In *FSE 2005*, pages 424-441. LNCS 3557, 2005.
- [17] F. Regazzoni, A. Cevrero, and F.-X. Standaert. A Design Flow and Evaluation Framework for DPA-Resistant Instruction Set Extensions. In *CHES 2009*, pages 205-219. LNCS 5747, 2009.
- [18] M. Renauld, F. X. Standaert, and N. V. Charvillon. Algebraic Side-Channel Attacks on the AES: Why Time also Matters in DPA. In *CHES 2009*, pages 99-111. LNCS 5747, 2009.
- [19] M. Saeki, D. Suzuki, K. Shimizu, and A. Satoh. A Design Methodology for a DPA-Resistant Cryptographic LSI with RSL Techniques. In *CHES 2009*, pages 189-204. LNCS 5747, 2009.
- [20] P. Schaumont and K. Tiri. Masking and Dual-Rail Logic Don't Add Up. In *CHES*

2007, pages 95-106. LNCS 4727, 2007.

[21] R. M. Sommer. Smartly Analyzing the Simplicity and the Power of Simple Power Analysis on Smart Cards. In CHES 2000, pages 78-92. LNCS 1965, 2000.

[22] D. Suzuki, M. Saeki, and T. Ichikawa. Random Switching Logic: A Countermeasure against DPA based on Transition Probability. <http://eprint.iacr.org> (2004).

[23] D. Suzuki, M. Saeki, and T. Ichikawa. DPA Leakage Models for CMOS Logic Circuits. In CHES 2005. LNCS 3659, 2005.

[24] K. Tiri and I. Verbauwhede. A Logic Level Design Methodology for a Secure DPA Resistant ASIC or FPGA Implementation. In Design, Automation and Test in Europe Conference (DATE 2004), pages 246-251, 2004.

[25] K. Tiri and I. Verbauwhede. Place and Route for Secure Standard Cell Design. In CARDIS 2004, pages 143-158. Kluwer Academic Publishers, 2004.

[26] N. Veyrat-Charvillon and F.-X. Standaert. Mutual Information Analysis: How, When and Why? In CHES 2009, pages 429-443. LNCS 5747, 2009.

[27] S. M. Yen and M. Joye. Checking Before Output May Not Be Enough Against Fault-Based Cryptanalysis. In IEEE Trans. on Computers, pages 967-970. IEEE 49, 2000.

[28] Y. Zhou and D. Feng. Side-channel Attacks: Ten Years After Its Publication and the Impacts on Cryptographic Module Security Testing. <http://eprint.iacr.org/2005/388/>, 2005.