Side Channel Attacks: Vulnerability Analysis of PRINCE and RECTANGLE using DPA

Ravikumar Selvam, Dillibabu Shanmugam, and Suganya Annadurai

Hardware Security Research Group,
Society for Electronic Transactions and Security, India.
{ravikumar,dillibabu,asuganya}@setsindia.net
http://www.setsindia.org/hardware.html

Abstract. Over a decade, cryptographers are more attentive on designing lightweight ciphers in focus to compact cryptographic devices. More often, the security of these algorithms are defined in terms of its resistance to mathematical cryptanalysis methods. Nevertheless, designers are well aware of implementation attacks and concentrating on new design strategies to improve the defence quality against implementation attack. PRINCE [3] and RECTANGLE [17] lightweight block ciphers are designed using new design strategies for efficiency and security. In this paper we analyse the security of PRINCE and RECTANGLE against a type of implementation attack called Differential Power Analysis (DPA) attack. Our attack reduces key search space from 2^{128} to 33008 for PRINCE and 2^{80} to 288 for RECTANGLE. To the best of our knowledge, this is the first DPA attack on PRINCE and RECTANGLE.
Keywords: Lightweight block cipher, power characteristic, FPGA implementation, differential power analysis

1 Introduction

Differential Power Analysis (DPA) attack, a type of implementation attack, exploits the power consumed by the device when it performs cryptography operations. In 1999, Kocher et al. [11] showed that power analysis attacks can efficiently reveal the secret key. After the DPA became public, designers of cryptographic algorithm had started concentrating on the new design strategies to improve the defense quality against the attack. However, few algorithms are still vulnerable to DPA attack. This motivated us to evaluate algorithms that are vulnerable to DPA attack.

Power analysis attack make use of the dynamic power consumption, which is the dominant factor in the total power consumption of the CMOS circuit[1]. The dynamic power consumption depends on the switching activity ($0 \rightarrow 1$ *or* $1 \rightarrow 0$) in the circuit. Thus, the power consumption is

[1] Generally, the physical devices used for cryptographic implementation are digital circuits, which are built based on CMOS process technology in practice.

dependent on the data that is processed by the cryptographic implementation. Then by measuring the power consumption during its operation an attacker may estimate the number of bit transitions in the device registers that are implemented using CMOS flip-flops. DPA attack makes use of the number of bit transitions that occurs during process of intermediate results. The number of bit transitions depends on the difference between previous and current state of the register.

With the advancement in FPGA technology, low-power FPGA [16] are expected to become popular for battery powered applications such as smartphones, RFID cards and wireless sensor nodes. Implementation of lightweight block ciphers on low-power FPGA have also been proposed recently [18]. Also, FPGAs are the preferred platform to investigate the power analysis vulnerabilities of cryptographic implementations, due to its low cost and flexibility [6]. For these reasons, we chose FPGA platform to practically verify our attack model.

Over a decade many new lightweight block ciphers such as HIGHT [9], DESL [12], PRESENT [2], KATAN [4], KLEIN [7], LED [8] have been proposed with various design strategies. These block ciphers are used to provide security for resource[2] constrained devices, termed as ubiquitous computing devices. Of these lightweight ciphers, two algorithms using new design strategy are taken for analysis: One is based on SPN with α-reflection property(`PRINCE`) and other is based on SPN with Bit-slice technique(`RECTANGLE`).

`PRINCE` is proposed in ASIACRYPT 2012 [3], by Julia Borghoff et. al. It is designed for low latency in hardware with features of reflectivity (single circuit for Encryption and decryption) and no real key schedule. The security analysis of `PRINCE` against Biclique and differential cryptanalysis, reflection cryptanalysis, differential fault attack and meet-in-the-middle attack were presented in [1,10,14,15,5,13]. In [17], `RECTANGLE` cipher is presented with it's security analysis. The cipher has bit-slice technique, which makes suitable for multiple platform with SCA resistant against timing and cache attacks. In this paper, we share the experimental results of the DPA attack on `PRINCE` and `RECTANGLE`.

Our contribution. In this paper, we present the power model chosen for power analysis attack on `PRINCE` and `RECTANGLE`. Then we present practically verified DPA attack on these algorithms using SASEBO-G board. Our attack reduces hypotheses complexity from 2^{128} to 33008 for

[2] Here resources include power, physical size of the device, computing capability and memory.

PRINCE and 2^{80} to 288 for RECTANGLE. To the best of our knowledge, this is the first DPA attack on PRINCE and RECTANGLE.

Outline. This paper is organised as follows. Section 2 describes, brief on PRINCE, its hardware implementation. In 3, power characteristics and DPA attack on physical implementation of PRINCE are presented. In Sect. 4, implementation details, power model and practical attack of RECTANGLE are discussed in detail. Finally we conclude the paper in Sect. 5.

2 Description of PRINCE

PRINCE has a block size of 64-bit, and key size of 128-bit. The 128-bit key is divided into two parts $K0\|K1$ each of 64-bit. The $K0'$ is derived from $K0$ as shown in (1).

$$K0' = (K0 \ggg 1) \oplus (K0 \gg 63) \tag{1}$$

Both $K0$ and $K0'$ are used as whitening keys. While the 64-bit key $K1$ is used in $PRINCE_{core}$, which is the 12-round iterative block cipher. The encryption process of PRINCE is depicted in Fig. 1.

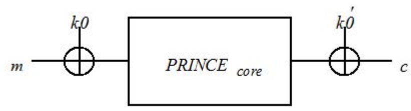

Fig. 1: PRINCE

The encryption process of $PRINCE_{core}$ is depicted in Fig. 2.

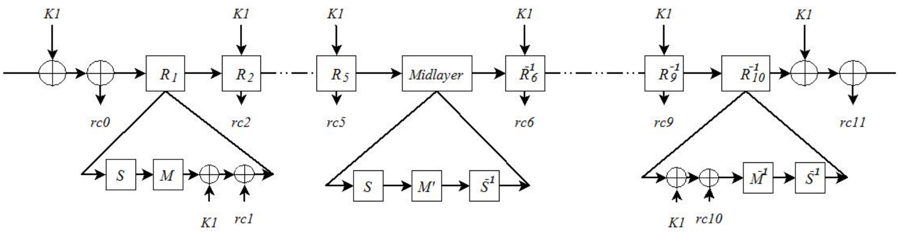

Fig. 2: $PRINCE_{core}$

The $PRINCE_{core}$ operates on a 4×4 column-major order matrix of nibbles, termed as state. Each round of $PRINCE_{core}$ is a combination of four operations, such as K_i-*add*, S-*layer*, M-*layer* and RC-*add*. The process of adding the key is termed as K_i-*add*, where the 64-bit key $K1$ is exclusive-ored with the state value.

S-*layer* represented as S, is a non-linear operation, where each nibble in the state is substituted by a nibble generated using the S-box. The author of `PRINCE` had recommended eight S-boxes to choose. We had chosen the S-box given in test vectors of `PRINCE` in [3]. The inverse S-box, represented as S^{-1}, which is also used in the encryption process from middle round onwards.

M-*layer* is a linear operation and is a composition of shift rows (SR) and M-mapping (M'). Shift rows (SR) permutes the 16 nibbles by rotating each row of state matrix by i cell positions to the left, where i varies from 0 to 3. The inverse of SR is denoted by SR^{-1}, which is the inverse permutation of nibbles. The matrix M' is a 64×64 matrix that is multiplied with the state matrix. The full description of formation of M' matrix is elaborated in [3]. The matrix M' has an involution property that is used in the middle round without shift rows. The inverse of M-*layer*, represented as M^{-1}, is the composition of M' and SR^{-1} and there is no inverse for M'.

RC-*add* is the add round constant operation, where the state matrix is exclusive-ored with the round constant. The pre-computed round constant values are given in [3].

2.1 Implementation in Hardware

The architecture of round-based implementation of `PRINCE`, as given in [3], is taken for our analysis. Here the round output refers to output of key whitening steps and $PRINCE_{core}$ round functions along with middle round. So that the `PRINCE` takes 15 clock cycles for one block of encryption. S-*layer* and M-*layer* have been implemented as boolean functions. Pre-computed round constants are realised as look-up table and RC-*add* fetches the constant value for its operation. In round based structure, a 64-bit register is used to store the intermediate result of each round output.

Initially, the 64-bit plaintext is loaded in `Register-64` and the register is updated for every clock cycle. Before entering into the $PRINCE_{core}$ block, the `Register-64` is updated by exclusive-or of plaintext and key ($K0$) as part of key whitening step. After completing the 12 rounds of $PRINCE_{core}$ operation, the state value of `Register-64` is exclusive-ored with key ($K0'$) that is derived from key ($K0$) as given (1). Thus the encryption completes and ciphertext is taken out from `Register-64`.

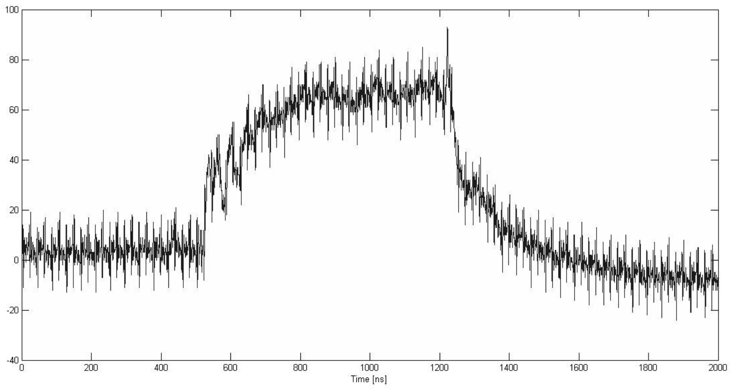

Fig. 3: Power trace for single encryption of `PRINCE`

The SASEBO-G board is used to experimentally verify the attack. Fig. 3 shows the power consumption of `PRINCE` during one encryption. It can be observed from Fig. 3 that the 15 patterns in the trace shows the encryption process of `PRINCE`. The pattern starts approximately at 585ns (nano second) and execution of each round leads to a pattern of approximately 41ns, while the FPGA board was operated at 24MHz frequency. `PRINCE` takes 615ns to complete its encryption process, which means that the pattern ends approximately at 1200ns. The trace points before and after the pattern are the power consumption during loading plain-text and cipher-text.

3 Power characteristics of PRINCE

Power analysis attack makes use of the number of bit transitions that occur during storage of intermediate results. In FPGA or ASIC implementation, intermediate results are stored in registers. The number of bit transitions of targeted intermediate result depend on the previous or next state of the register, which are assumed to be a known value. Therefore, Hamming distance power model is more suitable to estimate the number of bit transitions between states.

PRINCE algorithm is designed in such a way that the non-linear function[3] is used only in the second round of $PRINCE_{core}$. Key whitening function and first round of $PRINCE_{core}$ are designed using exclusive-or operation. Hence the second round output is the targeted intermediate result for DPA. To target second round function, both $K0$ and $K1$ should be known to find the previous state value of target state. But, in PRINCE, the previous and next state of targeted intermediate result are unknown. The attack perception of PRINCE is elaborated as follows

$$P = [p_0 \ p_1 \ p_2 .....p_{15}]; \tag{2}$$

$$K0 = [k0_0 \ k0_1 \ k0_2 ....k0_{15}]; \tag{3}$$

$$K1 = [k1_0 \ k1_1 \ k1_2 ....k1_{15}]; \tag{4}$$

Equation (2) represents the 64-bit plaintext as 16 4-bit elements. Similarly, (3) and (4) represent the keys $K0$ and $K1$ respectively. In PRINCE algorithm, the key whitening step is included to increase the attack complexity twice. Nevertheless, this became advantageous for us to do single hypothesis for Key (K) which is exclusive-or of $K0$ and $K1$. This can be defined as

$$K = [k_0 \ k_1 \ k_2 ....k_{15}]; \tag{5}$$

where

$$K = K0 \oplus K1; \tag{6}$$

In PRINCE, the unknown previous state and their values of the register are represented as T^j and α_{index} respectively. Here *index* represents position of elements placed in the state.

$$T^j = \begin{pmatrix} \alpha_0 & \alpha_4 & \alpha_8 & \alpha_{12} \\ \alpha_1 & \alpha_5 & \alpha_9 & \alpha_{13} \\ \alpha_2 & \alpha_6 & \alpha_{10} & \alpha_{14} \\ \alpha_3 & \alpha_7 & \alpha_{11} & \alpha_{15} \end{pmatrix}$$

The state values of T^j is first round output of $PRINCE_{core}$, where key $K1$ and round constant $rc0$ are exclusive-or with result of key whitening step. The round constant for first round is all zeros, so that the state value is exclusive-or of plaintext and key K. Equation (7),(8),(9), and

[3] In DPA, the non-linear function helps to uniquely determine the correct key guess, even if a key hypothesis is wrong in only one bit.

(10) shows the state values with respect to plaintext and key K.

$$\alpha_0[3] = p_0[3] \oplus k_0[3]; \tag{7}$$

$$\alpha_0[2] = p_0[2] \oplus k_0[2]; \tag{8}$$

$$\alpha_0[1] = p_0[1] \oplus k_0[1]; \tag{9}$$

$$\alpha_0[0] = p_0[0] \oplus k_0[0]; \tag{10}$$

The targeted state is the result of second round and is represented as T^{j+1} and the values of T^{j+1} is represented as β_{index}.

$$T^{j+1} = \begin{pmatrix} \beta_0 & \beta_4 & \beta_8 & \beta_{12} \\ \beta_1 & \beta_5 & \beta_9 & \beta_{13} \\ \beta_2 & \beta_6 & \beta_{10} & \beta_{14} \\ \beta_3 & \beta_7 & \beta_{11} & \beta_{15} \end{pmatrix}$$

The second round of $PRINCE_{core}$, passes previous state value through round function operation such as S-*layer*, M-*layer*, and exclusive-or of key $K1$ and round constant $rc1$. Each bit of T^{j+1} can be written as follows:

$$\beta_0[3] = S_1[3] \oplus S_2[3] \oplus S_3[3] \oplus k1_0[3] \oplus rc1_0[3]; \tag{11}$$

$$\beta_0[2] = S_0[2] \oplus S_2[2] \oplus S_3[2] \oplus k1_0[2] \oplus rc1_0[2]; \tag{12}$$

$$\beta_0[1] = S_0[1] \oplus S_1[1] \oplus S_3[1] \oplus k1_0[1] \oplus rc1_0[1]; \tag{13}$$

$$\beta_0[0] = S_0[0] \oplus S_1[0] \oplus S_2[0] \oplus k1_0[0] \oplus rc1_0[0]; \tag{14}$$

where,
$S_0 = S\text{-}layer(\alpha_0)$; $S_1 = S\text{-}layer(\alpha_1)$;
$S_2 = S\text{-}layer(\alpha_2)$; $S_3 = S\text{-}layer(\alpha_3)$;

Here we describe the targeted intermediate output, by taking single nibble element and explained its bit dependency with the previous state value and key bits. In M-*layer*, M' is a 64×64 matrix, each row contains only three $1's$. This activate single bit of three different nibbles on the same column. S-*layer* takes four bit input and give four bit output by diffusing every bit of input to all bits of output. Due to this the dependency of an output bit gets raised from 3-bits to 3-nibbles.

In equation (11), the previous state elements $\alpha_1,\ \alpha_2,\ and\ \alpha_3$ along with key $k1_0[3]$ are required, for finding single bit output of $\beta_0[3]$. The previous state value containing the corresponding key bits as $k_1,\ k_2,\ and\ k_3$. Similarly each bit of targeted intermediate output depends on 12-bits of key K and 1-bit from key $K1$ of corresponding position.

3.1 Pragmatic Execution

PRINCE algorithm is practically implemented and executed for 30,000 random plaintext with fixed key $K0 = [DF8B\ A07C\ 946B\ 5E13]$ and $K1 = [698B\ 31E5\ F06B\ 4629]$. The power traces are captured, with 4000 trace points per encryption of 30,000 samples. The key recovery has been structured by taking the suitable elements of all the plaintexts and exclusive-or with 12-bit key hypothesis. This created the hypothetical previous state value of targeted bit. These values are then operated for second round function of $PRINCE_{core}$ with additional single-bit hypothesis of key $K1$. Totally 13-bits have been hypotheses to estimate both state values. For example in equation (7) and (11), the plaintext nibbles $p_1\ p_2\ p_3$ are exclusive-or with the key hypothesis $k_1\ k_2\ k_3$ and the result is given to non-linear layer as part of second round function. As stated by *M-layer* most significant bit of every nibble is taken out from the output of *S-layer* to perform exclusive-or operation. The resultant bit is again exclusive-ored with a hypothetical key bit $k1_0[3]$ and round constant $rc1_0[3]$ as given in (11). Now the Hamming distance calculated between most significant bit of α_3 and β_3 is given by.

$$HD(\alpha_0[3], \beta_0[3]) = HW(\alpha_0[3] \oplus \beta_0[3]) \tag{15}$$

Likewise, Hamming distance of $\alpha_0[2]$ and $\beta_0[2]$ obtained by picking the plaintext $[p_0\ p_1\ p_3]$ and key-bits $[k_0\ k_1\ k_3]$ with $k1_0[2]$ are performed. Both the results are shown in Fig. 4 and Fig. 5. Same procedure is applied on each bit of the state value to reveal the complete 64-bit key K with complexity of 2^{13} per bit and also reveals key $K1$ of targeted position. The key $K0$ has been recovered by XOR-ing obtained $K1$ and K.

Fig. 4 and Fig. 5 show the plot between correlation value and 13-bit key hypotheses. In Fig. 4, both positive and negative peaks are having the same correlation value of 0.03111 and the 13-bit key values are at 1536 ([0 6 0 0] in Hexadecimal)and 5633 ([1 6 0 0]). Both the peaks are having same 12-bit values ([6 0 0]) and differ only in the most significant bit. The most significant bit is single-bit hypothesis of $k1_0[3]$ and remaining 12-bit are key K ($[k_1\ k_2\ k_3]$). The single-bit hypothesis of $k1_0[3]$ results

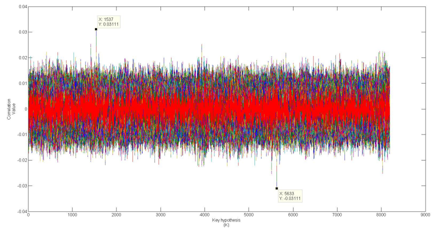

Fig. 4: key recovery of $\beta_0[3]$

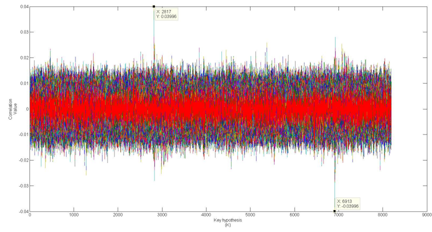

Fig. 5: key recovery of $\beta_0[2]$

$'0'$, which gives the positive peak at 1536 is correct key guess. This occurs due to the targeted key bit from $K1$ is exclusive-ored with result of M-*layer*,that is after non-linear function. Hence the complement of targeted key bit leads to negative correlation. Similarly from Fig. 5, the highest correlation value 0.03996 stands at 2817 and 6913 as positive and negative values respectively. The targeted intermediate result depends on 12-bit key element $[k_0\ k_2\ k_3]$ are revealed as $[B\ 0\ 0]$ and single bit $k1_0[2]$ as $'0'$ for 2816. Thus the two bits on each nibble elements of any column reveals the complete 64-bit key K with complexity of $2^{13} \times 2^2 + 2^5 \times 2^2 = 2^{15} + 2^7$ and also reveals the 8-bit of $K1$. The remaining 56- bits of $K1$ can be recovered by reusing the obtained key K and doing single-bit hypothesis for the remaining position of $K1$. After revealing the complete $K1$, the key $K0$ is obtained by exclusive-or of $K1$ and K without any additional complexity. So the overall complexity of revealing the 128-bit key is about $2^{15} + 2^7 + 112 = 33008$.

4 Description of RECTANGLE

RECTANGLE is a lightweight block cipher, designed using bit-slice technique; which makes the cipher adoptable for multiple platforms (Hardware and Software). It has round function of 25 iterations with 64-bit block length and 80- or 128-bit key length. Each round function consists of the following three steps: AddRoundKey, SubColumn and ShiftRow. After final round, output is exclusive-or with the final round key.

Initialization (P/R^i): Stores Plain-text(P) or Intermediate(R^i) values.

AddRoundKey(ARK)**:** A round subkey is bitwise exclusive-or with intermediate state.

SubColumn(SC)**:** 4-bit SBoxes are executed in parallel on the column of the state. Sbox is tabulated as below Table 4.

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S(x)	9	4	F	A	E	1	0	6	C	7	3	8	2	B	5	D

ShiftRows(SR)**:** First row has no shift, Second row is left rotated by 1-bit position, third row is left rotated by 12-bit positions and the last row is left rotated by 13-bit positions.

RECTANGLE, is implemented in SASEBO-G board using Verilog Hardware Descriptive Language. The power consumption of the cipher is as shown in Fig. 6.

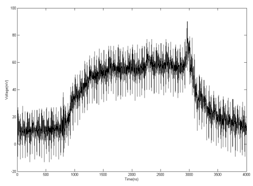

Fig. 6: Power trace of RECTANGLE

From the Fig. 6 it is clear that the power consumption is high for 25 iterations and then comes down.

4.1 Description of power model and attack

Power model should realistically describe power consumption between the intermediate states of the algorithm in hardware module. Each round has four state, such as Initialization(Pliantext/Intermediate), AddRoundkey, SubColumn, ShiftRows as shown in 7. In order to do implementation(DPA) attack, their should be a diffusion of key bits over non-linear function. In our case, sub-column has the property of non-linear function. Each round executes 16 sub-columns in parallel on its corresponding inputs, which has key bit influences. These bits diffuse over the column by the function. Then, shift row is a permutation operation has no significant influence in power consumption. Therefore, each column of the (R^i) is targeted as intermediate value. To model the power consumption of the round function, Hamming distance is ideal; which use to correlate column of the plaintext(P) with corresponding column of the intermediate state(R^i) or between two successive round function intermediate states, say, (R^1) and (R^2).

By this 64-bit key is revealed from 16 columns between plaintext state (P) and the first round intermediate state (R^1). For remaining 16 key bits, the first round R^1 is correlated with corresponding bits in second round R^2 as shown in Table 1.

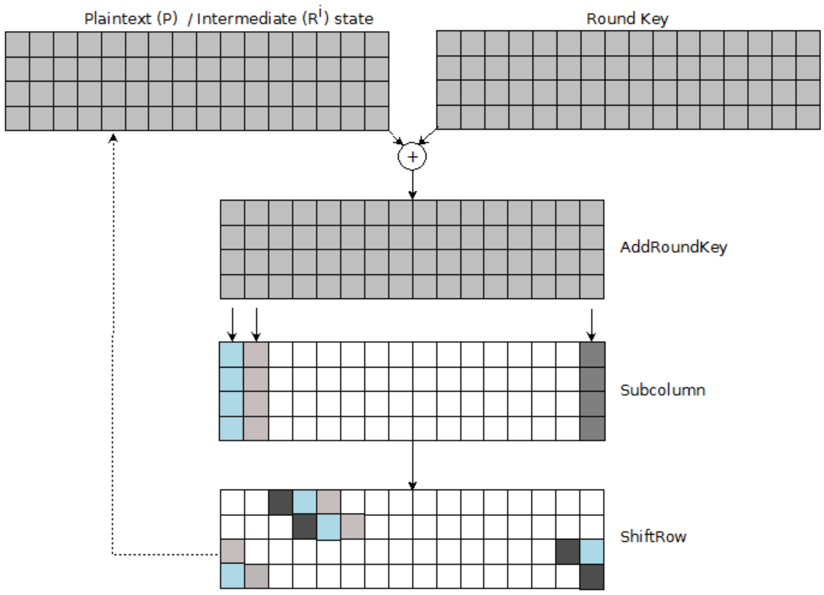

Fig. 7: `RECTANGLE` Round Function

$$HD = HW(P \oplus R^i) \tag{16}$$

R^i is an intermediate state value.
where, $R^1 = SR(SC(P \oplus K))$

Power consumption of intermediate state(R^i) columns is given by the below equation.

$$Power_{column} = (P_{0,j} \oplus R^1_{0,j}) + (P_{1,j} \oplus R^1_{1,j}) + (P_{2,j} \oplus R^1_{2,j}) + (P_{3,j} \oplus R^1_{3,j}) \tag{17}$$

where $0 \leqslant i \leqslant 3$ and $0 \leqslant j \leqslant 15$ refers indices of the row and column of the R^i state register.

4.2 Summary of our attack

The power consumption of `RECTANGLE` is captured while it encrypts `D` (with minimum of 2,00,000 samples) randomly generated plaintexts using a fixed key K[4].

Intermediate hypothetical value is calculated for D plaintexts using R^1. Hypothetical power consumption value is calculated by taking Hamming distance between P and R^1 as given in equation 17. Then the hypothetical power consumption value is correlated with the actual power consumption value. For example, first column of P and R^1 is correlated as shown Fig. 8 for 4-bit key hypothesis. The peak appears at 10, this means the correct key is 9 (because the index for key hypothesis in the plot starts from 1) with the correlation value of 0.01185. The correct key and its corresponding bit positions are given below. Similarly, second column of P and R^1 is correlated and its key values are given in Fig. 9. This process is repeated for sixteen columns of R^1 to reveal 64-bit key. The remaining key bits are retrieved by correlating R^1 and R^2. Complete 80 key bits recovery and its corresponding bits correlation with attack complexity is tabulated as below 1.

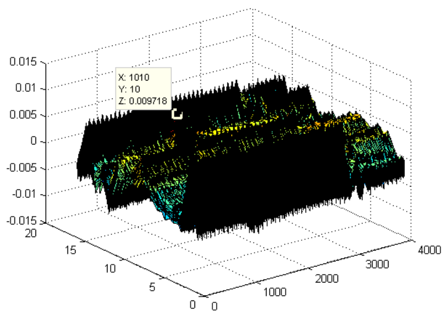

Fig. 8: First column correlation between P and R^1: $[P_{48}, P_{32}, P_{16}, P_0; R^1_{61}, R^1_{44}, R^1_{17}, R^1_0]$

K^1_{60}	K^1_{40}	K^1_{20}	K^1_0
1	0	0	1

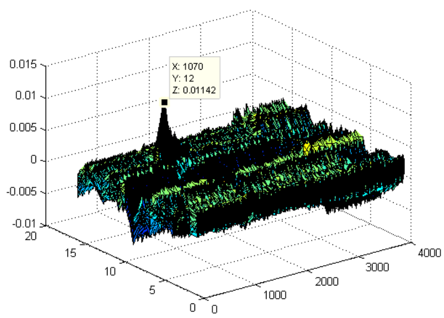

Fig. 9: Second column correlation between P and R^1: $[P_{49}, P_{33}, P_{17}, P_1; R^1_{62}, R^1_{45}, R^1_{18}, R^1_1]$

K^1_{61}	K^1_{41}	K^1_{21}	K^1_1
1	0	1	1

4 The key K that was used for experiment is K = [6 9 8 7 B 4 A 5 F 0 D 2 3 C E 1 5 A 4 B], represented in hexadecimal.

Attack complexity. Differential Power analysis(DPA) is divide and conquer approach. That is, instead of trying brute force approach to reveal 80-bit key with complexity of 2^{80}; chunks of key bits are attacked with reduced complexity. Therefore attack complexity is significantly reduced from 2^{80} to $(2^4 * 16) + (2^2 * 6) + (2^1 * 4) = 288$. By the same way, other variants of RECTANGLE algorithm can also be attacked using DPA.

Table 1: Attack complexity of RECTANGLE

Correlation between states	Correlation bits	Attack bits(Key)	Hypothesis
	$[P_{48}, P_{32}, P_{16}, P_0; R^1_{61}, R^1_{44}, R^1_{17}, R^1_0]$	$K_{60}, K_{40}, K_{20}, K_0$	16
	$[P_{49}, P_{33}, P_{17}, P_1; R^1_{62}, R^1_{45}, R^1_{18}, R^1_1]$	$K_{61}, K_{41}, K_{21}, K_1$	16
	$[P_{50}, P_{34}, P_{18}, P_2; R^1_{63}, R^1_{46}, R^1_{19}, R^1_2]$	$K_{62}, K_{42}, K_{22}, K_2$	16
	$[P_{51}, P_{35}, P_{19}, P_3; R^1_{48}, R^1_{47}, R^1_{20}, R^1_3]$	$K_{63}, K_{43}, K_{23}, K_3$	16
	$[P_{52}, P_{36}, P_{20}, P_4; R^1_{49}, R^1_{32}, R^1_{21}, R^1_4]$	$K_{64}, K_{44}, K_{24}, K_4$	16
	$[P_{53}, P_{37}, P_{21}, P_5; R^1_{50}, R^1_{33}, R^1_{22}, R^1_5]$	$K_{65}, K_{45}, K_{25}, K_5$	16
	$[P_{54}, P_{38}, P_{22}, P_6; R^1_{51}, R^1_{34}, R^1_{23}, R^1_6]$	$K_{66}, K_{46}, K_{26}, K_6$	16
	$[P_{55}, P_{39}, P_{23}, P_7; R^1_{52}, R^1_{35}, R^1_{24}, R^1_7]$	$K_{67}, K_{47}, K_{27}, K_7$	16
	$[P_{56}, P_{40}, P_{24}, P_8; R^1_{53}, R^1_{36}, R^1_{25}, R^1_8]$	$K_{68}, K_{48}, K_{28}, K_8$	16
P, R^1	$[P_{57}, P_{41}, P_{25}, P_9; R^1_{54}, R^1_{37}, R^1_{26}, R^1_9]$	$K_{69}, K_{49}, K_{29}, K_9$	16
	$[P_{58}, P_{42}, P_{26}, P_{10}; R^1_{55}, R^1_{38}, R^1_{27}, R^1_{10}]$	$K_{70}, K_{50}, K_{30}, K_{10}$	16
	$[P_{59}, P_{43}, P_{27}, P_{11}; R^1_{56}, R^1_{39}, R^1_{28}, R^1_{11}]$	$K_{71}, K_{51}, K_{31}, K_{11}$	16
	$[P_{60}, P_{44}, P_{28}, P_{12}; R^1_{57}, R^1_{40}, R^1_{29}, R^1_{12}]$	$K_{72}, K_{52}, K_{32}, K_{12}$	16
	$[P_{61}, P_{45}, P_{29}, P_{13}; R^1_{58}, R^1_{41}, R^1_{30}, R^1_{13}]$	$K_{73}, K_{53}, K_{33}, K_{13}$	16
	$[P_{62}, P_{46}, P_{30}, P_{14}; R^1_{59}, R^1_{42}, R^1_{31}, R^1_{14}]$	$K_{74}, K_{54}, K_{34}, K_{14}$	16
	$[P_{63}, P_{47}, P_{31}, P_{15}; R^1_{60}, R^1_{43}, R^1_{16}, R^1_{15}]$	$K_{75}, K_{55}, K_{35}, K_{15}$	16
	$[R^1_{51}, R^1_{35}, R^1_{19}, R^1_3; R^2_{48}, R^2_{47}, R^2_{20}, R^2_3]$	K_{16}	2
	$[R^1_{52}, R^1_{36}, R^1_{20}, R^1_4; R^2_{49}, R^2_{32}, R^2_{21}, R^2_4]$	K_{17}	2
	$[R^1_{53}, R^1_{37}, R^1_{21}, R^1_5; R^2_{50}, R^2_{33}, R^2_{22}, R^2_5]$	K_{36}, K_{18}	4
	$[R^1_{54}, R^1_{38}, R^1_{22}, R^1_6; R^2_{51}, R^2_{34}, R^2_{23}, R^2_6]$	K_{37}, K_{19}	4
R^1, R^2	$[R^1_{55}, R^1_{39}, R^1_{23}, R^1_7; R^2_{52}, R^2_{35}, R^2_{24}, R^2_7]$	K_{56}, K_{38}	4
	$[R^1_{56}, R^1_{40}, R^1_{24}, R^1_8; R^2_{53}, R^2_{36}, R^2_{25}, R^2_8]$	K_{57}, K_{39}	4
	$[R^1_{57}, R^1_{41}, R^1_{25}, R^1_9; R^2_{54}, R^2_{37}, R^2_{26}, R^2_9]$	K_{76}, K_{58}	4
	$[R^1_{58}, R^1_{42}, R^1_{26}, R^1_{10}; R^2_{55}, R^2_{38}, R^2_{27}, R^2_{10}]$	K_{77}, K_{59}	4
	$[R^1_{59}, R^1_{43}, R^1_{27}, R^1_{11}; R^2_{56}, R^2_{39}, R^2_{28}, R^2_{11}]$	K_{78}	2
	$[R^1_{60}, R^1_{44}, R^1_{28}, R^1_{12}; R^2_{57}, R^2_{40}, R^2_{29}, R^2_{12}]$	K_{79}	2
Combined hypothesis			288

5 Conclusion

The results show that `PRINCE` and `RECTANGLE` cipher are vulnerable to DPA attack and requires additional scheme to secure over the practical attack. Our work extends to incorporate countermeasure and analyse the effect of countermeasure against higher order attacks. Also we plan to analyse the overhead introduced by the countermeasure and to explore the possible optimisation techniques.

References

1. Farzaneh Abed, Eik List, and Stefan Lucks. On the security of the core of prince against biclique and differential cryptanalysis. Cryptology ePrint Archive, Report 2012/712, 2012. http://eprint.iacr.org/.
2. Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, Yannick Seurin, and C. Vikkelsoe. PRESENT: An Ultra-Lightweight Block Cipher. In Pascal Paillier and Ingrid Verbauwhede, editors, *CHES*, volume 4727 of *Lecture Notes in Computer Science*, pages 450–466. Springer, 2007.
3. Julia Borghoff, Anne Canteaut, Tim Güneysu, Elif Bilge Kavun, Miroslav Knezevic, Lars R. Knudsen, Gregor Leander, Ventzislav Nikov, Christof Paar, Christian Rechberger, Peter Rombouts, Søren S. Thomsen, and Tolga Yalçin. Prince - a low-latency block cipher for pervasive computing applications - extended abstract. In Xiaoyun Wang and Kazue Sako, editors, *ASIACRYPT*, volume 7658 of *Lecture Notes in Computer Science*, pages 208–225. Springer, 2012.
4. Christophe De Cannière, Orr Dunkelman, and Miroslav Knezevic. KATAN and KTANTAN - A Family of Small and Efficient Hardware-Oriented Block Ciphers. In Christophe Clavier and Kris Gaj, editors, *CHES*, volume 5747 of *Lecture Notes in Computer Science*, pages 272–288. Springer, 2009.
5. Anne Canteaut, Maria Naya-Plasencia, and Bastien Vayssire. Sieve-in-the-middle: Improved mitm attacks (full version). Cryptology ePrint Archive, Report 2013/324, 2013. http://eprint.iacr.org/.
6. Cryptography Research Inc. Protecting FPGAs from Power Analysis. http://www.cryptography.com/public/pdf/FPGASecurity.pdf. Accessed: July, 2014.
7. Zheng Gong, Svetla Nikova, and Yee Wei Law. KLEIN: A New Family of Lightweight Block Ciphers. In Ari Juels and Christof Paar, editors, *RFIDSec*, volume 7055 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2011.
8. Jian Guo, Thomas Peyrin, Axel Poschmann, and Matthew J. B. Robshaw. The LED Block Cipher. In Bart Preneel and Tsuyoshi Takagi, editors, *CHES*, volume 6917 of *Lecture Notes in Computer Science*, pages 326–341. Springer, 2011.
9. Deukjo Hong, Jaechul Sung, Seokhie Hong, Jongin Lim, Sangjin Lee, Bonseok Koo, Changhoon Lee, Donghoon Chang, Jaesang Lee, Kitae Jeong, Hyun Kim, Jongsung Kim, and Seongtaek Chee. HIGHT: A New Block Cipher Suitable for Low-Resource Device. In Louis Goubin and Mitsuru Matsui, editors, *CHES*, volume 4249 of *Lecture Notes in Computer Science*, pages 46–59. Springer, 2006.
10. Jérémy Jean, Ivica Nikolic, Thomas Peyrin, Lei Wang, Shuang Wu, et al. Security analysis of prince. 2013.

11. Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael J. Wiener, editor, *CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.
12. Gregor Leander, Christof Paar, Axel Poschmann, and Kai Schramm. New Lightweight DES Variants. In Alex Biryukov, editor, *FSE*, volume 4593 of *Lecture Notes in Computer Science*, pages 196–210. Springer, 2007.
13. Leibo Li, Keting Jia, and Xiaoyun Wang. Improved meet-in-the-middle attacks on aes-192 and prince. Cryptology ePrint Archive, Report 2013/573, 2013. http://eprint.iacr.org/.
14. Hadi Soleimany, Céline Blondeau, Xiaoli Yu, Wenling Wu, Kaisa Nyberg, Huiling Zhang, Lei Zhang, and Yanfeng Wang. Reflection cryptanalysis of prince-like ciphers. *Journal of Cryptology*, pages 1–27, 2013.
15. Ling Song and Lei Hu. Differential fault attack on the prince block cipher. Cryptology ePrint Archive, Report 2013/043, 2013. http://eprint.iacr.org/.
16. T. Tuan, A. Rahman, S. Das, S. Trimberger, and Sean Kao. A 90-nm Low-Power FPGA for Battery-Powered Applications. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 26(2):296–300, 2007.
17. Wentao Zhang, Zhenzhen Bao, Dongdai Lin, Vincent Rijmen, Bohan Yang, and Ingrid Verbauwhede. Rectangle: A bit-slice ultra-lightweight block cipher suitable for multiple platforms. Cryptology ePrint Archive, Report 2014/084, 2014. http://eprint.iacr.org/.
18. Xueying Zhang, Howard M. Heys, and Cheng Li. FPGA Implementation and Energy Cost Analysis of Two Light-Weight Involutional Block Ciphers Targeted to Wireless Sensor Networks. *MONET*, 18(2):222–234, 2013.