SCADPA: Side-Channel Assisted Differential-Plaintext Attack on Bit Permutation Based Ciphers

Jakub Breier, Dirmanto Jap and Shivam Bhasin
Physical Analysis and Cryptographic Engineering
Temasek Laboratories at Nanyang Technological University
Singapore
Email: {jbreier, djap, sbhasin}@ntu.edu.sg

Abstract—**Bit permutations are a common choice for diffusion function in lightweight block ciphers, owing to their low implementation footprint. In this paper, we present a novel Side-Channel Assisted Differential-Plaintext Attack (SCADPA), exploiting specific vulnerabilities of bit permutations. SCADPA is a chosen-plaintext attack, knowledge of the ciphertext is not required. Unlike statistical methods, commonly used for distinguisher in standard power analysis, the proposed method is more differential in nature. The attack shows that diffusion layer can play a significant role in distinguishing the internal cipher state. We demonstrate how to practically exploit such vulnerability to extract the secret key. Results on microcontroller-based PRESENT-80 cipher lead to full key retrieval using as low as 17 encryptions. It is possible to automate the attack by using a thresholding method detailed in the paper. Several case studies are presented, using various attacker models and targeting different encryption modes (such as CTR and CBC). We provide a discussion on how to avoid such attack from the design point of view.**

I. Introduction

Lightweight cryptography is an emerging branch of cryptology that has gained importance with the rise of internet of things. Algorithms for lightweight cryptography are designed to work in resource-constrained environments like low area footprint, low energy etc. More specifically, lightweight cryptography is designed to address security concerns on low-cost platforms. Recently, *NIST* has launched a call for proposal to standardize lightweight crypto algorithms [1]. This work focuses on lightweight block ciphers which use bit permutation based diffusion layer to achieve efficient implementations. Common examples of such block ciphers are PRESENT [2], which is an *ISO* standard, GIFT [3], etc.

Bit permutation is an interesting design choice as it has negligible area footprint in hardware and can be implemented with only wires [2]. Other ciphers, such as GIFT [3], with careful permutation choices can be optimized for both hardware and software implementations.

In this paper, we bring forward a specific vulnerability of bit permutation based diffusion function, which can be simply exploited using side-channel. The exploit arises from the simple structure of bit permutation and is not easily found in diffusion functions of standard ciphers (like `MixColumns` in AES). The proposed vulnerability is more serious in low-cost platforms like 8-bit microcontrollers due to serialized execution of the algorithm. Such design vulnerabilities further make the need of countermeasures critical, however, lightweight and countermeasures do not often go hand-in-hand. Cipher designers generally add extra rounds to avoid vulnerabilities due to simple diffusion layer. Since the proposed attack exploits all the information in the first round, extra rounds will not add any security.

A bit permutation layer diffuses the output bits of an Sbox (non-linear Substitution layer) to multiple Sboxes. By observing the numbers of affected Sboxes in a given round, the (key-dependent) Sbox output in the previous round can be determined, thus revealing information about the secret key. In this paper, we present Side-Channel Assisted Differential-Plaintext Attack (SCADPA) which exploits bit permutation construction for secret key retrieval through observed side-channel leakage. Here, SCADPA is chosen plaintext attack, where the plaintexts are chosen to effectively exploit the bit-permutation leakage. It observes the difference of propagation through side-channel, thus revealing the differential of Sbox output. With the knowledge of the plaintext, this differential can be solved to reveal the corresponding key candidates.

Unlike usual side-channel attacks (SCA [4]), SCADPA is not a statistical attack but rather a differential attack. For a carefully chosen set of plaintexts, it observes differential on an internal sensitive value. As we show later, these differentials for bit permutation ciphers can be obtained by simply subtracting the power measurements. Once the internal differentials are known, the attack is similar to classical differential cryptanalysis for secret key retrieval. While cryptanalysis can be applied on a full cipher and handle high complexity, we manage to restrict the attack to a single round, thus keeping the complexity negligible. In some cases, multiple plaintext pairs might be needed to determine a unique key candidate. With the demonstrated experiments on PRESENT-80, SCADPA can reveal the key in 17 encryptions for the best case and 65 in the worst case.

A. Related Works

Chosen plaintext side-channel attacks have previously been proposed in different context. A side-channel based collision attack [5] was proposed under chosen plaintext setting. It detects collision in some internal value of initial rounds of the cipher to retrieve the key. This attack was extended to break secret AES-like ciphers [6] and further generalised to SPN structures [7]. Some proposed attacks also used chosen plaintext setting to amplify power [8] or timing [9] side-channel leakage. Apart from block ciphers, chosen plaintexts attack were also applied to break public key cryptography [10] and hash functions (HMAC [11]). To the best of our knowledge, SCADPA is the first attack proposition to exploit the diffusion function in a block cipher.

B. Our Contribution

The contributions of this paper are as follows:

- We identify a specific vulnerability in bit-permutation based diffusion functions exploitable through side-channel, and we propose a specific attack called SCADPA, which exploits the identified vulnerability.
- We present a practical demonstration of SCADPA on a low cost platform, using PRESENT-80 lightweight cipher as a target algorithm.
- We analyze the efficiency of SCADPA for several scenarios, such as multiple nibbles retrieval, different block cipher modes of operation, etc.
- We provide analysis of the best, average and worst case on the recovered key and the number of encryptions required, highlighting the effectiveness of the proposed method.
- We further explain why such vulnerabilities might be absent in other diffusion layer construction.

C. Organization

The rest of the paper is organised as follows. Sec. II first recalls basics of PRESENT-80 block cipher, and then describes the key methodology of SCADPA. Experimental validation of SCADPA on an 8-bit microcontroller is presented in Sec. III. Some discussions are provided in Sec. IV, exploring attack on cipher operation mode, vulnerability of other diffusion function etc. Final conclusions are drawn in Sec. V.

II. Methodology

In this section, we first provide an overview of PRESENT cipher, while detailing the properties of this algorithm that we exploit. Later, we explain how the SCADPA method works and finally, we present a simple example. Although the rest of the paper is based on PRESENT-80, the techniques are generally applicable on similar ciphers.

A. PRESENT Cipher

PRESENT is a lightweight block cipher based on Substitution-Permutation Network (SPN) [2]. Therefore, it consists of three operations: addRoundKey is a bit-wise xor of the state with the round key; sBoxLayer is a nibble-wise nonlinear substitution; pLayer is a bit permutation. The structure of one round of the cipher is depicted in Fig. 1. PRESENT consists of 31 rounds, followed by a post-whitening addRoundKey at the end. The variant used in our experiments, PRESENT-80, has a secret key of size 80 bits and a block size of 64 bits. Tab. I shows PRESENT Sbox that is executed on all 16 nibbles of the PRESENT-80 state during the sBoxLayer. The Sbox function is further denoted as $S(.)$.

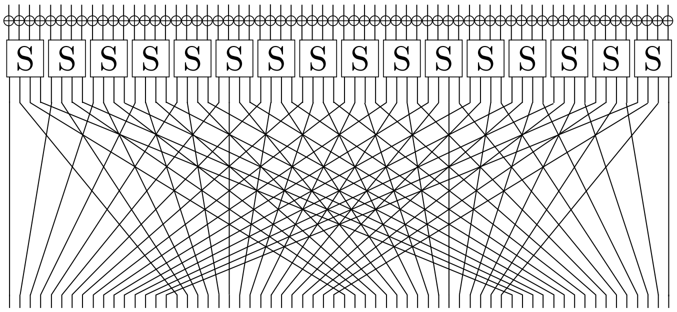

Fig. 1: Structure of one round of PRESENT-80 cipher.

TABLE I: PRESENT S-box.

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S(x)$	C	5	6	B	9	0	A	D	3	E	F	8	4	7	1	2

B. Bit Permutation Properties of PRESENT-80

There are three main properties of the pLayer that are exploited in the following:

1) Output of one nibble is distributed into four distinct nibbles.
2) Input to one nibble consists of outputs from four distinct nibbles.
3) In pLayer, the cipher state can be split into four different groups of four nibbles, where one input group affects exactly one output group.

Examining these properties, it can be seen that by changing chosen four nibbles in the plaintext, it is possible to affect the whole cipher state after the first permutation.

Fig. 2 shows this behavior by changing the first and the eighth nibble of the plaintext. The underlying implementation computes sBoxLayer nibble-wise while addRoundKey is computed byte-wise owing to the ALU support for bit-wise xor. This is to achieve the best speed-memory trade-off. Similarly, in 32-bit architectures, sBoxLayer would be implemented as a 4-bit or 8-bit look-up table and addRoundKey with pLayer would be done on 32-bit words, to achieve best speed-memory trade-off.

By observing the changing nibbles in round 2, the change in Sbox output at round 1 can be determined. This value directly depends on the secret key which can be exploited for key retrieval. In the following, we use side-channel measurements to observe the changed nibbles.

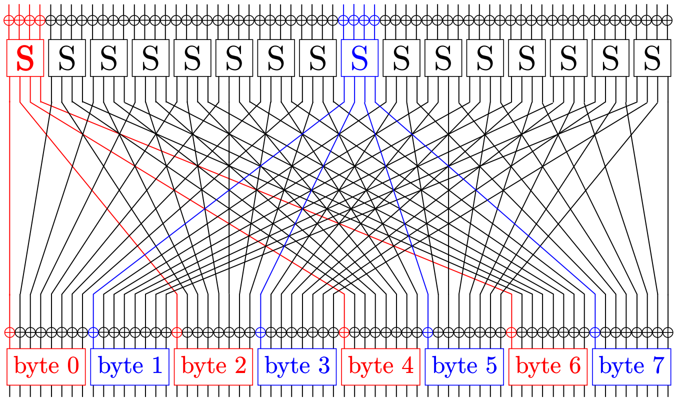

Fig. 2: Bytes in round 2 that could be potentially affected by changing the first and the eighth nibble of the plaintext.

C. SCADPA Methodology

Using the information from the previous part, we propose SCADPA. SCADPA exploits the permutation properties to observe changed nibbles and retrieve differential at Sbox output in round 1. The differential can be solved for key retrieval by using a standard differential attack on non-linear layer.

The attack steps can be summarized as follows:

Step 1: Encrypt a chosen plaintext p, by using an unknown secret key k, denoted as $E_k(p)$.

Step 2: Capture the power/EM leakage of the Device Under Test (DUT) during the second encryption round to get the trace t.

Step 3: Choose another plaintext $p' \neq p$ that differs exactly in one nibble at i^{th} position. The nibble at position i in plaintext p is denoted p_i and $x_i = p_i \oplus k_i$. Similarly, $x'_i = p'_i \oplus k_i$.

Step 4: Capture the leakage for $E_k(p')$ to get t'.

Step 5: Calculate $\Delta t = t - t'$.

Step 6: By examining Δt, get the Sbox output change $\Delta S_i = S(x_i) \oplus S(x'_i)$ of round 1 in the position where the plaintext had changed.

Step 7: From ΔS_i and (p_i, p'_i), calculate all possible candidates for key nibble k_i (satisfying $S(p_i \oplus k_i) \oplus S(p'_i \oplus k_i) = \Delta S_i$).

Step 8: Repeat steps 3-7 with another p'_i, taking intersection of all the calculated key candidates, until there is just one candidate for k_i.

Step 9: Repeat steps 3-8 for all $i \in [1, 16]$ to recover the whole round key.

Step 10: Compute the remaining 16 bits of the secret key by exhaustive search or repeating SCADPA on the next round.

D. SCADPA Acceleration by Optimal Plaintext Choice

In order to reduce the key complexity and retrieve the secret key with fewer number of encryptions, it is possible to change the value of multiple nibbles in the plaintext. Based on the permutation layer, multiple nibbles can be changed without affecting same locations in the next round and hence, can be analyzed independently. The optimal methodology executes the following steps:

Step 1: Keep the record of nibbles of key that have not been recovered ($I = \{1, ..., 16\}$)

Step 2: Start by choosing one nibble ($n_i \in I$) and calculate all possible affected nibbles at the beginning of the next round (N_i).

Step 3: Choose another nibble ($n_j \in I$, $n_j \neq n_i$) and check if the affected nibbles interfere (check if $N_i \cap N_j = \emptyset$). If true, keep n_j, else, move to other nibble . Repeat, until no nibble remaining, then update $I \setminus \{n_i, n_j, ...\}$.

Step 4: Choose plaintext set that changes only on these nibble positions ($\{n_i, n_j, ...\}$).

Step 5: Repeat step 2-4, until $I = \emptyset$

Another option is to choose the pair difference in the plaintext that could minimize the key candidate. This is dependent on the Sbox used. In the case of PRESENT, as shown later in Figure 5, for one plaintext pair, there could be either 2 or 4 key candidates. However, even with two plaintext pairs, there is still slight chance on getting more than 1 key candidate.

A natural question would be – ‘how many nibbles of PRESENT can we attack at once by using SCADPA?’ As can be seen in Fig. 2, one nibble can affect up to half of the state at the next round `addRoundKey`. Nibbles $0-7$ (“group 1”) affect bytes $0, 2, 4, 6$, while nibbles $8-15$ (“group 2”) affect the remaining bytes $1, 3, 5, 7$. Therefore, by combining one nibble from each group, we can retrieve two nibbles at the same time while avoiding the interference. This knowledge can help us to reduce the number of encryptions to half. Parallelisation of attack to two nibbles is limited by the byte-wise granularity of `addRoundKey`. The following `sBoxLayer` with nibble-wise granularity would allow up to 4 nibbles in parallel, however at the cost of needed profiling.

E. Attack Example

We illustrate our method by a simple example that shows how to recover one nibble of the round key i.e. $i = 1$. We fix $p_i =$ `0x0` and $p'_i =$ `0xA`. After observing the Δt, we figure out that $\Delta S_i =$ `0x2`. By knowing that $\Delta p_i =$ `0xA`, there can be two candidates for k_i: `0xF` and `0x5`. We make another experiment, now with $p'_i =$ `0x3`. By capturing another trace, we determine $\Delta S_i =$ `0x6`, giving us four different candidates for k_i: `0xC`, `0xD`, `0xE`, and `0xF`. The only intersecting candidate is `0xF`, therefore we know that the key nibble has to be this value. The retrieval of ΔS_i from real power traces is explained in the following section.

III. Experimental Results

In this section, we will show and discuss experimental results obtained by performing SCADPA on a microcontroller implementation of PRESENT-80 cipher [2].

A. Setup

As a DUT, we used a standard 8-bit microcontroller from Atmel, ATmega328P, mounted on Arduino UNO development board. We have measured an electromagnetic emanation with a Langer RF-U 5-2 probe. Signal was captured with LeCroy WaveRunner 610 Zi oscilloscope.

We have used an implementation of PRESENT-80 cipher, where `sBoxLayer` is computed nibble-wise and the rest of the operations are done byte-wise. Sampling rate was set at 500 MS/s, while the `addRoundKey` takes ≈7000 samples. A difference introduced in the first round could be observed during the second round. We chose to observe the difference at the second `addRoundKey` as in our case the start of round was easily identifiable by visual inspection of the trace. Choice of observing the difference on `addRoundKey` has an advantage and a disadvantage. The advantage being that precise profiling was not needed as compared to locating time instants for `sBoxLayer`. The disadvantage is that since `addRoundKey` is done byte-wise, the observed differences are limited to byte level. Observed differences on the following `sBoxlayer` can be nibble precise, given appropriate profiling.

B. Results

To support our method, we have conducted experiments showing the possibility of distinguishing ΔS.

Fig 3 shows differences in power consumption captured in the second `addRoundKey`, by calculating Δt. The implementation we used computes the `addRoundKey` in a reverse order, therefore the difference peaks follow this order. In order to improve signal-to-noise ratio and produce clear plots, both t and t' were averaged from multiple executions. Nevertheless, it was possible to see the difference and raw traces. By observing this difference, the output difference of `sBoxLayer` in round 1, i.e. ΔS_i, can be clearly distinguished. Once ΔS_i and Δp_i are known, it can be used to solve the value of secret key k_i, as shown in the previous section.

C. Automatic Recognition of the Difference

If we look at Fig. 3, it is obvious that determination of differences can be automated. For this purpose, one can use a simple algorithm that sets the threshold, enabling the separation of interesting areas from the random noise. Following steps can be done in order to make the automatic difference recognition:

Step 1: Calculate the mean (μ) and the standard deviation (σ) from the difference trace Δt, however both traces t and t' come from the same plaintext, i.e. there is no difference in the Sbox output.

Step 2: Set the basic positive and negative threshold to be $r_+ = \mu + \sigma$ and $r_- = \mu - \sigma$, respectively.

Step 3: Choose n to be a multiplier of r_+ and r_- in a following way: start from $n = 1$ and increment the value by 1 until all the points of the trace fall between the positive and negative threshold.

Step 4: Use these values for all the measured traces to indicate points where the difference traces cross the thresholds.

Step 5: By the density distribution of the crossing points, together with the timing, determine ΔS.

Thresholds for $\Delta S =$ `0x01` are stated in Fig. 4.

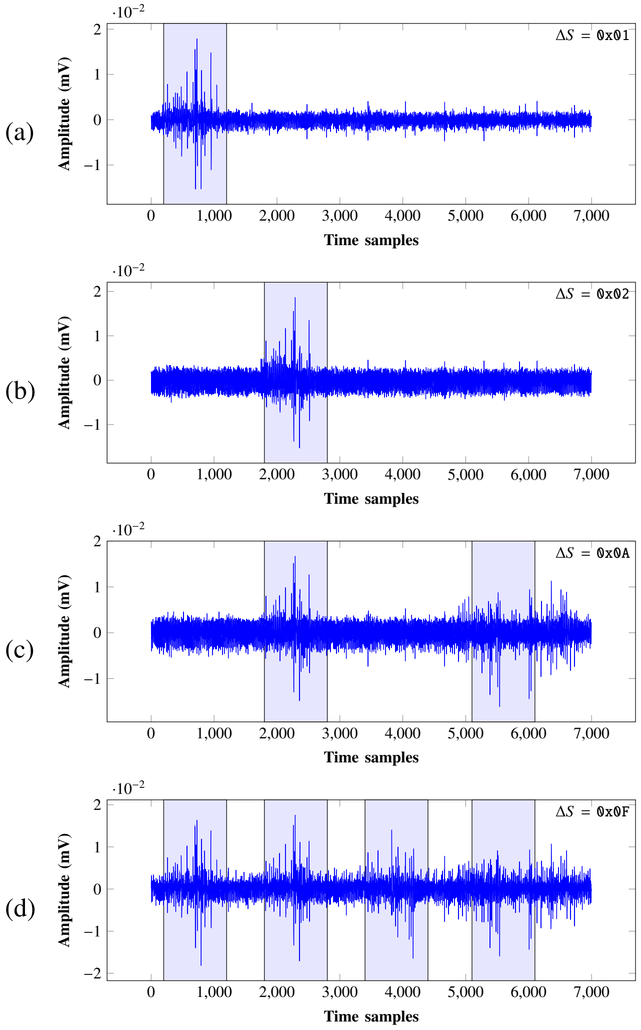

Fig. 3: Difference traces Δt showing `addRoundKey` of round 2, revealing the output difference of the Sbox at round 1. Bytes are processed in a reversed order, therefore the pattern showing the difference also has to be reversed. Difference between the Sbox outputs for the plots (highlighted by the blue background) is as follows: (a) `0x01`, (b) `0x02`, (c) `0x0A`, and (d) `0x0F`.

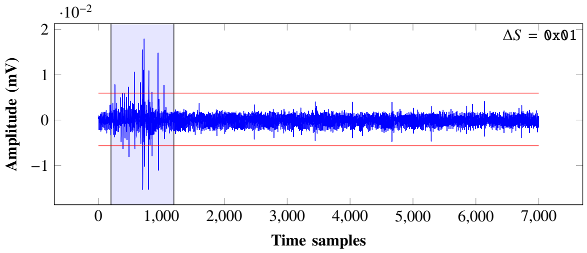

Fig. 4: Threshold that enables determination of the ΔS. Values used for thresholding were: $\mu = 3.452 \times 10^{-5}, \sigma = 1.453 \times 10^{-3}, n = 4$.

TABLE II: Probability of determining key candidates.

	# of plaintext pairs		
# of key candidates	1	2	3
1	0	0.94315	0.99557
2	0.59948	0	0
3	0	0	0
4	0.40052	0.05685	0.00443

D. Attack Complexity

In this section, we compute the attack complexity for single key nibble retrieval and full round key retrieval.

The single nibble retrieval complexity depends on the underlying Sbox function, being the only non-linear function in the derived differential equation. As can be seen in Tab. II, using a single plaintext pair $\Delta p'_i$ for determining ΔS, it will yield 2 key candidates in $\approx 60\%$ of cases and 4 candidates for the rest. By adding additional plaintext pair, the probability of identifying a unique key candidate k_i are 94.3%, 99.5% and 100% for 2, 3 and 4 plaintext pair respectively. Since, the same reference plaintext can be used, with 5 encryptions a unique key candidate can be identified in worst case. The numbers would change for an Sbox other than PRESENT, but will stay comparable.

Now we compute the number of encryptions required for full key retrieval. In the best case, difference in two nibbles in the plaintexts will results in 8 byte difference in the next round, with no interference. With the appropriate plaintext values, 3 values are sufficient to reduce the key complexity to 1, resulting to unique key solution. Thus, in the best case, only 17 encryptions are required for recovering the whole round key, as indicated in Fig. 5. In case we can still choose the optimal plaintexts but for some reason can only recover one nibble at a time, it will require 33 encryptions. The worst case only recovers one nibble at a time while using the conservative reduction of candidates and not exploiting any specific Sbox properties. In this case, it will require 4 values to reduce the key candidate to 1 per nibble. Hence, it will require 65 encryptions.

Please note that it is also possible to make a search on remaining candidates. For example, if we have 2 candidates for each nibble, we can determine the value with a brute-force search with complexity of 2^{16}. For such case, it would only require 9 encryptions in case we target two nibbles at a time. Similarly, operating at nibble wise granularity at the following `sBoxLayer` can recover the key in 9 encryptions as well.

IV. Further Discussion

A. Attack on Different Modes of Operation

Few block cipher operation mode are well oriented towards plaintext selection of SCADPA. When it comes to Counter (CTR) mode (Fig. 6), the input to the encryption algorithm consists of a nonce and a counter. While the nonce is a random number, counter normally increases by 1 after each block. While the nonce stays fixed, the incrementing counter satisfies the chosen plaintext criteria of SCADPA as discussed in Sec. II. The attack allows to recover few nibbles directly affected by the counter in the first round. For the remaining nibbles, chosen nonce or attack on second round can be carried out in a similar way.

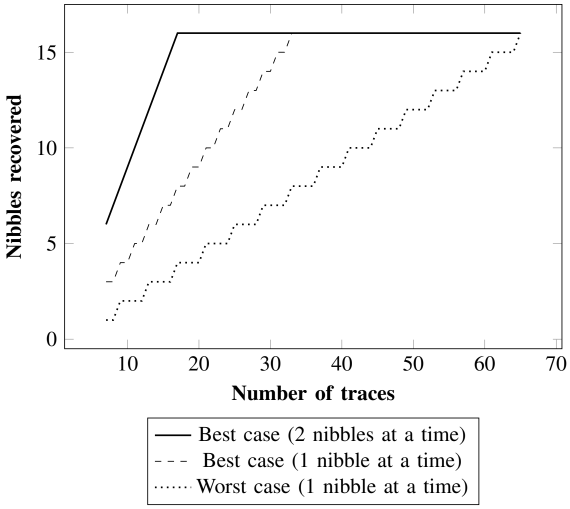

Fig. 5: Number of traces required for recovering key nibbles of PRESENT.

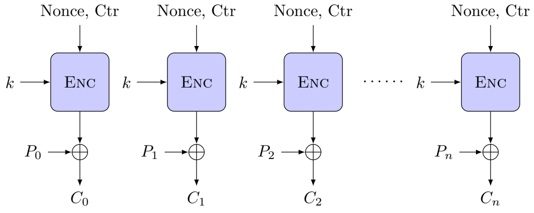

Fig. 6: Counter (CTR) mode of operation (encryption) [12].

On the other hand, when targeting Cipher Block Chaining (CBC) mode, one has to aim at the decryption module (Fig. 7). This comes from the property of CBC where the plaintext is first `xor`-ed with the IV (first block) or with ciphertext from the previous block. In this case, chosen-plaintext attack changes to chosen-ciphertext, without the knowledge of plaintext. The same hold for Propagating Cipher Block Chaining (PCBC) mode.

B. Alternatives for Diffusion Layer

Certain diffusion function do not offer vulnerabilities that are exploited by SCADPA in bit permutation. AES [13], the *NIST* standard for symmetric key cryptography, is a relevant

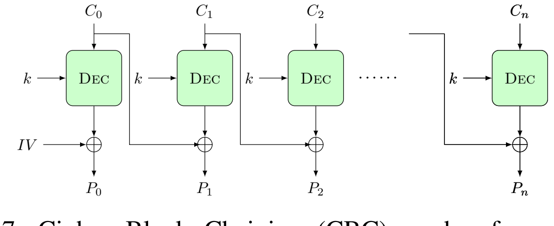

Fig. 7: Cipher Block Chaining (CBC) mode of operation (decryption) [12].

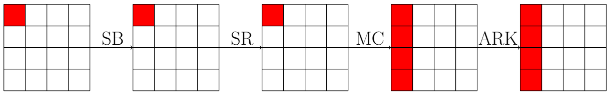

Fig. 8: Difference Diffusion in AES.

example. AES encrypts 128-bit data block with a 128/192/256 bits secret key in 10/12/14 rounds. The data is organised in a 4×4 matrix of bytes called state and the round function is applied upon it. A round comprises of four operations i.e. `SubBytes` (SB), `ShiftRows` (SR), `MixColumns` (MC) and `AddRoundKeys` (ARK). SB is a 8×8 non-linear table look up and ARK is the round key addition. We concentrate on diffusion functions i.e. SR and MC. SR applies a cyclic shift on the rows (1,2,3,4) with offsets (0,1,2,3). MC operates on four bytes of each column of the state. The four bytes are combined using an invertible linear transformation. When a difference is inserted at the input, irrespective of the plaintext difference, the difference is always propagated on all four bytes, preventing ΔS leakage. This is illustrated in Fig. 8.

Recent trends show that it is possible to design a lightweight cipher with similar diffusion function and not only rely on bit permutations. SKINNY [14], PRINCE [15] are common examples. Other ultra-lightweight ciphers like SIMON and SPECK [16], use several bit shifts applied to a partial intermediate state to provide the diffusion. Furthermore, only a binary operation is used to provide non-linearity. Both operations combined would prevent a successful application of SCADPA.

C. A Note on Countermeasures

As SCADPA exploits leakage of bit permutation through side-channel, any countermeasure which prevents direct observation of side-channel information can protect against such attacks. This includes both hiding [17] and masking countermeasures [18]. However, any bias can still render the attack possible. For instance, an imbalanced implementation of hiding will still allow to observe the difference. Similarly for masking, ΔS would depend upon $p_i \oplus m_i \oplus p'_i \oplus m'_i$. If mask m_i, m'_i are totally independent and uniformly distributed, the attack is not possible, however with biases in the mask, the attack can still be carried out with increased effort. Shuffling of the order of the Sboxes and/or key additions would make the attack harder since only the hamming weight could be directly observed, instead of the difference value.

As previously discussed, countermeasures incur significant overheads. When considering lightweight cryptography specially oriented for low-cost platforms, implementation weaknesses as shown in bit permutation can be avoided.

V. Conclusions

In this paper, we identify a vulnerability in bit permutation based lightweight ciphers (PRESENT, GIFT, etc) and develop a side-channel assisted methodology called SCADPA to exploit it. With a practical case study on low-cost microcontroller running PRESENT-80, we were able to practically recover the secret key with as low as 17 encryptions and an exhaustive search with complexity of 2^{16}. Several attacker models are presented, with different complexities of retrieving the key. Use of more complex yet low-cost diffusion function are encouraged to avoid such vulnerabilities.

In the future work, it would be interesting to look at possibilities of exploiting different side-channel countermeasures. Especially, if randomness in masking is biased or the leakage characteristics of hiding are not uniform.

References

[1] K. A. McKay, L. Bassham, M. S. Turan, and N. Mouha, "Report on lightweight cryptography," *NIST DRAFT NISTIR*, vol. 8114, 2016.

[2] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. Robshaw, Y. Seurin, and C. Vikkelsoe, "PRESENT: An ultra-lightweight block cipher," in *CHES*, vol. 4727. Springer, 2007, pp. 450–466.

[3] S. Banik, S. K. Pandey, T. Peyrin, Y. Sasaki, S. M. Sim, and Y. Todo, "GIFT: A Small Present," *Cryptographic Hardware and Embedded Systems-CHES*, pp. 25–28, 2017.

[4] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Advances in cryptology - CRYPTO'99*. Springer, 1999, pp. 789–789.

[5] K. Schramm, T. Wollinger, and C. Paar, "A new class of collision attacks and its application to DES," in *FSE*, vol. 2887. Springer, 2003, pp. 206–222.

[6] C. Clavier, Q. Isorez, and A. Wurcker, "Complete SCARE of AES-Like Block Ciphers by Chosen Plaintext Collision Power Analysis," in *INDOCRYPT*, vol. 8250. Springer, 2013, pp. 116–135.

[7] M. Rivain and T. Roche, "SCARE of secret ciphers with SPN structures," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2013, pp. 526–544.

[8] N. Veyrat-Charvillon and F.-X. Standaert, "Adaptive chosen-message side-channel attacks." in *ACNS*, vol. 6123. Springer, 2010, pp. 186–199.

[9] J. Kelsey, B. Schneier, D. Wagner, and C. Hall, "Side channel cryptanalysis of product ciphers," *Computer Security - ESORICS 98*, pp. 97–110, 1998.

[10] B. den Boer, K. Lemke, and G. Wicke, "A DPA Attack against the Modular Reduction within a CRT Implementation of RSA," in *CHES*, vol. 2523. Springer, 2002, pp. 228–243.

[11] L. Guo, L. Wang, D. Liu, W. Shan, Z. Zhang, Q. Li, and J. Yu, "A chosen-plaintext differential power analysis attack on HMAC-SM3," in *Computational Intelligence and Security (CIS), 2015 11th International Conference on*. IEEE, 2015, pp. 350–353.

[12] J. Jean, "TikZ for Cryptographers," https://www.iacr.org/authors/tikz/, 2016.

[13] N. F. Pub, "197: Advanced encryption standard (AES)," *Federal information processing standards publication*, vol. 197, no. 441, p. 0311, 2001.

[14] C. Beierle, J. Jean, S. Kölbl, G. Leander, A. Moradi, T. Peyrin, Y. Sasaki, P. Sasdrich, and S. M. Sim, "The SKINNY Family of Block Ciphers and its Low-Latency Variant MANTIS," Cryptology ePrint Archive, Report 2016/660, 2016, http://eprint.iacr.org/2016/660.

[15] J. Borghoff, A. Canteaut, T. Güneysu, E. B. Kavun, M. Knezevic, L. R. Knudsen, G. Leander, V. Nikov, C. Paar, C. Rechberger *et al.*, "PRINCE–a low-latency block cipher for pervasive computing applications," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2012, pp. 208–225.

[16] R. Beaulieu, S. Treatman-Clark, D. Shors, B. Weeks, J. Smith, and L. Wingers, "The SIMON and SPECK lightweight block ciphers," in *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, June 2015, pp. 1–6.

[17] P. Hoogvorst, J.-L. Danger, and G. Duc, "Software Implementation of Dual-Rail Representation," in *COSADE*, 2011, Darmstadt, Germany.

[18] E. Prouff and M. Rivain, "Masking against side-channel attacks: A formal security proof," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2013, pp. 142–159.