J.Natn.Sci.Foundation Sri Lanka 2017 45 (4): 393 - 404
DOI: http://dx.doi.org/10.4038/jnsfsr.v45i4.8233

RESEARCH ARTICLE

Breaking Speck cryptosystem using correlation power analysis attack

Hasindu Gamaarachchi*, Harsha Ganegoda and Roshan Ragel
Department of Computer Engineering, Faculty of Engineering, University of Peradeniya, Peradeniya.

Revised: 01 April 2017; Accepted: 20 July 2017

Abstract: This is an empirical study based on correlation power analysis for extracting the secret key from a Speck based embedded cryptosystem. Speck was recently introduced by the National Security Agency, USA as a lightweight software-based block cipher targetting embedded systems. The pervasive nature of embedded devices makes it crucial to perform this assessment. We empirically show that the secret key of a Speck cryptosystem can be extracted within an hour on a microcontroller based embedded system. The experiments were performed in three different microcontrollers with different bit widths and power settings. It is concluded that although Speck is new, it is vulnerable to power analysis attack.

Keywords: Correlation power analysis (CPA), cryptography, differential power analysis (DPA), power analysis attack, Speck.

INTRODUCTION

Power analysis attacks extract the secret key of a cryptosystem by investigating the power consumption of the device when the system is running. This has become a threat to the security of embedded devices such as smart cards (Mangard *et al*., 2007). They target embedded systems that can come under the physical possession of the attacker, but yet the damage caused can be significant. According to Eisenbarth *et al*. (2008), it has been possible to break KeeLoQ remote key-less entry systems used in garage openers and car doors using power analysis in few minutes, making it possible to clone the remote control. Liu *et al*. (2015) have recently shown that the secret key of modern 3G/4G USIM cards can be recovered in few minutes paving the way to eavesdrop, clone cards and bypass one time password authentication. Power analysis also allows the secret key of television set-top boxes to be recovered. This allows the attacker (customer in this scenario) to freely access the premium content without subscription (Rambus Corporation, 2016). Further, many devices such as swipe card door locks, credit cards, encrypted flash drives, single sign-on smartcards and bit-locker smartcards can be targets of power analysis attacks.

Advanced encryption standard (AES), the most popular block cipher at present can be attacked within 10 minutes using this method (Gamaarachchi *et al*., 2015) on a cryptosystem with no countermeasures implemented. Power analysis attacks on algorithms such as AES and data encryption standard (DES), exploit the substitution box (S-Box, a precomputed table that makes the software implementation of such algorithm faster) lookup as the point of interest to measure power (Zhu *et al*., 2013). Therefore, a large number of countermeasures have been proposed to solely make the S-Box lookups secure.

Speck (Beaulieu *et al*., 2013) is a block cipher introduced by the National Security Agency (NSA), USA in 2013, which is optimised for software implementations. It is a lightweight cipher with very low memory requirements and processing power. Hence it is ideal for software based cryptosystems in embedded devices. Further, with the advancements of internet of things (IoT), the number of embedded systems that

* Corresponding author (hasindu2008@gmail.com; https://orcid.org/0000-0002-9034-9905)

require cryptographic operations are rapidly increasing. Speck would be an ideal candidate for such devices as stated in Beaulieu *et al*. (2015). Therefore, there is a considerable chance of Speck being widely used in the future. Hence, evaluating the security of Speck is crucial. In this paper, an empirical study based on correlation power analysis (CPA) attack (Brier *et al*., 2004) for extracting the secret key from a Speck based embedded cryptosystem is presented.

The immunity of a Speck cryptosystem against power analysis attacks might be overestimated due to the following notions:

- Power analysis attacks on traditional block ciphers exploit S-Box lookups. However, Speck does not have an S-Box.
- Power analysis attacks are old techniques, and Speck is a recently introduced block cipher. A newer algorithm might have been designed not to be vulnerable to previously discovered attacks.
- Lookup operations, such as the one in AES's S-Box are performed byte by byte. The three operations of Speck, *add*, *rotate* and *xor* utilise all the bits in the datapath of a processor. At a given moment, the power consumption of a Speck cryptosystem will correspond to that of all the bits in the datapath.

This paper shows that although attacking Speck is more challenging than attacking AES, still it is possible. While elaborating the challenges and the ways to overcome them, we derive that even on a microcontroller with 64-bit datapath an attack would be feasible. Further, we practically demonstrate that on a 8-bit microcontroller, Speck could be attacked in about half an hour and on a 16-bit microcontroller, in about one hour. Thus, we emphasise that it is unsafe to underestimate the necessity of countermeasures for Speck based cryptosystems.

Power analysis attack

The dynamic power consumption of complementary metal oxide semiconductor (CMOS) technology based electronic devices depends on the data that is being processed and operations being performed by the devices (Mangard *et al*., 2007). Therefore, it is possible to recover the secret key of a cryptosystem by measuring and analysing the power consumption of the device during encryption/decryption.

The general model of a power analysis attack can be explained as follows. A host computer sends the plaintext samples to the cryptographic device. Power traces are recorded onto the computer by using an oscilloscope while the device is under operation. This process is repeated for a multiple number of plaintext samples to acquire multiple power traces. The elaborated attack is a known plaintext attack, but power analysis attack can also be launched as a known ciphertext attack (Kocher *et al*., 1999). The collected power traces are then analysed using techniques such as simple power analysis (SPA) and differential power analysis (DPA) (Kocher *et al*., 1999). Running the analysis would return the secret key of the system.

Correlation power analysis (CPA)

CPA (Brier *et al*., 2004) is a variant of DPA. It is a statistical technique based on Pearson correlation. It works even on noisy power measurements, while still requiring a less number of power traces compared to the other techniques (Brier *et al*., 2004). Therefore, we used CPA for the analysis. A traditional brute force attack on a 128-bit key cipher would be practically infeasible even on a super computer as there are 2^{128} key possibilities to test. On the other hand in a CPA attack, the key is attacked byte by byte separately, where there are only 256×16 combinations. This is a much smaller and feasible time complexity when compared to the brute force attack. In a cryptosystem with a software implementation of AES, byte by byte attack is inherently possible because the S-Box lookups are performed byte-wise and the power consumption at a given moment is that of a single byte (Mangard *et al*., 2007).

CPA is done focusing on an intermediate value that is calculated during the encryption. The intermediate value should be a result of a function (known as the selection function) of key and plaintext (or ciphertext). In AES, usually it is the value after the S-Box lookup in the first round. The hypothetical power consumption of the circuit when reading or writing the considered intermediate value can be mathematically computed using a power model such as the Hamming weight model. Hypothetical power values are computed for all possible key combinations of a keybyte for each plaintext sample used. Then these hypothetical power consumption values are compared with the real power values in the acquired power traces by using Pearson correlation. The keyguess with the highest correlation is selected as the correct keybyte. The process is repeated for all the keybytes.

Let N number of plaintext samples be used to capture N number of power traces. Let each power trace has M number of sampling points with respect to time. The j^{th} sample point of the i^{th} power trace is written as $W_{i,j}$. The hypothetical power consumption value for the i^{th} plaintext with respect to a certain keyguess (for a single keybyte) is written as H_i. An estimate for the correlation coefficient

for j^{th} sampling point for the considered keybyte is found using equation (1).

$$\hat{\rho} = \frac{N\sum_{i=0}^{N} W_{i,j}H_i - \sum_{i=0}^{N} W_{i,j}\sum_{i=0}^{N} H_i}{\sqrt{N\sum_{i=0}^{N} W_{i,j}^2 - (\sum_{i=0}^{N} W_{i,j})^2}\sqrt{N\sum_{i=0}^{N} H_i^2 - (\sum_{i=0}^{N} H_i)^2}} \quad ...(1)$$

Speck

Speck (Beaulieu *et al*., 2013) only consists of three less compute intensive operations, namely *add*, *rotate*, and *xor*. Speck does not have a S-Box as in AES and hence the memory requirement is low. The source code of Speck is also small that it consumes a small programme memory. For example, when an AES implementation (encryption only) in C is around 300 lines of code, the same for Speck is about 20 lines. Hence, Speck could be implemented even on a very low end microcontroller. Therefore, it is an ideal candidate for software based embedded devices that require cryptographic operations.

Different combinations of block and key sizes make ten variants of Speck. With the increase of the key size, the number of encryption rounds also increase. The smallest variant is with 32-bit block size, 64-bit key size, and 22 rounds. The largest is with 128-bit block size, 256-bit key size, and 34 rounds.

Figure 1 depicts a block diagram of the first two rounds of Speck. Despite the variant of Speck, the rest of the rounds are very similar. The top of Figure 1 shows the transformation of plaintext. The plaintext is first divided into two parts namely *PT*1 and *PT*2. The round key generation process is at the bottom of Figure 1. The key is also first split into two halves namely *K*1 (left half key) and *K*2 (right half key). The round keys are generated using *add*, *rotate* and *xor* operations on *K*1 and *K*2. Plaintext segments *PT*1 and *PT*2 are jumbled using *add*, *rotate* and *xor* operations and mixed with round keys using *xor* operations as shown. With the small size and simple operations, Speck has become an ideal candidate for block cipher based encryption in embedded systems.

There is a large number of work in literature for performing power analysis attacks on block ciphers such as AES (Martinasek *et al*., 2013; Petrvalsky *et al*., 2013; 2014; Gamaarachchi *et al*., 2014) and DES (Messerges *et al*., 1999; Clavier *et al*., 2000).

The approaches for attacking previous cryptographic algorithms such as AES and DES are not suitable for a Speck cryptosystem due to the reasons stated previously. There are several works that perform differential cryptanalysis on Speck. Differential cryptanalysis attacks are based on the resultant differences on the output caused by differences in the input. A conventional differential attack and a rectangular attack on Speck is presented in Abed *et al*. (2014). Work such as Biryukov *et al*. (2014) and Dinur (2014) also perform similar but improved differential attacks on Speck. Further, Liu *et al*. (2016) have presented a linear cryptanalysis attack on variants of Speck with block sizes of 32, 48, 64, 96 and 128 bits.

Work on side channel attacks targetting Speck is very limited. A differential fault analysis attack on Speck has been presented in Tupsamudre (2014) that reveals the internal state of the cryptosystem by injecting faults. According to the best of our knowledge, currently there is no work on attacking Speck by using power as the side channel. Hence, we present an empirical study on how the key of a Speck cryptosystem can be derived by using CPA, which is a variant of power analysis attacks.

In this paper, we show that extracting the secret key from a Speck cryptosystem is still possible, even though it is a challenge when compared to the attack on AES. Therefore, our contributions in this paper are:

(1) a methodology, including the selection function, presented for breaking Speck, and
(2) the empirical results presented for Speck's vulnerability to power analysis attack.

METHODOLOGY

In the Speck algorithm, the secret key is not directly mixed with the plaintext as it is in AES. Instead, only the right half of the key, *K*2, is directly mixed as shown in Figure 1. The left half of the key (*K*1) is used for generating round keys that are subsequently mixed in next rounds. Further, the S-Box lookups are used in AES as intermediate values for the attack, but Speck does not have such. Hence, the approach used for breaking AES does not work for Speck.

We present an approach to break Speck, which is shown in Figure 2a. The attack is done in two separate phases. First, the right half of the key is recovered. Then the left half key is recovered with the help of the already recovered right half key.

Attacking the right half key

First, the power traces must be captured such that those include the power consumption of the device during the first round. In a laboratory environment, this is done by

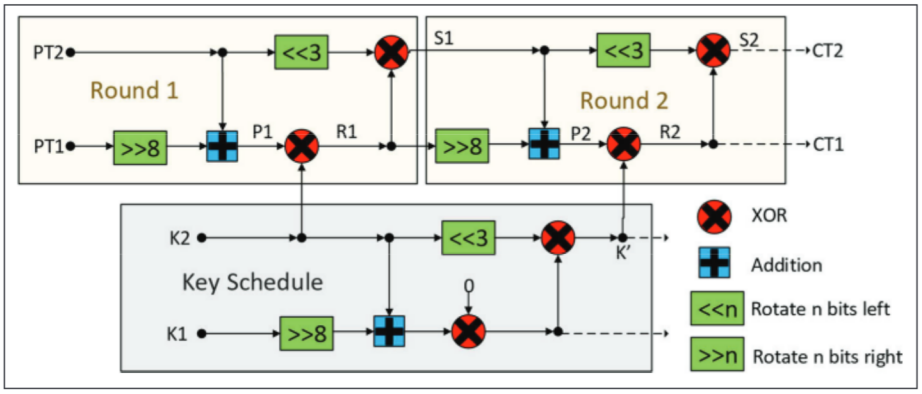

Figure 1: Block diagram of Speck [*https://en.wikipedia.org/wiki/Speck_(cipher)*]

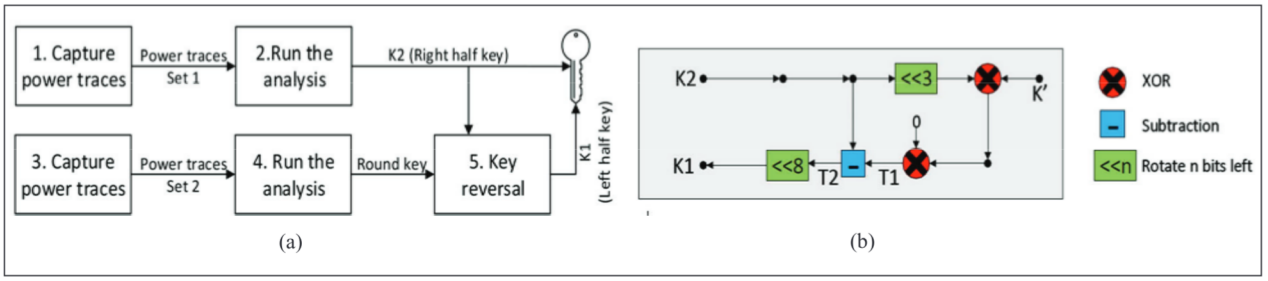

Figure 2: (a) Phases of the attack ; (b) key reversal process

setting a trigger based on a manually programmed pin that remains high during the required period. In a real attack situation, the moment at which the communication of plaintext stop can be used as a trigger. Since instruction cycle time is known and fixed, the part of the power trace until the required point can be then trimmed off. Even the beginning of communication can be used as a trigger for the acquisition. If both the power trace and communication line are captured, the required part can be later cropped out by scanning the point, in which the fluctuations in the communication line finishes.

The next step as shown in Figure 2(a) is the correlation power analysis of power traces to recover $K2$. The value $R1$ after the *xor* operation between the modified plaintext $P1$ and $K2$ (Figure 1) is selected as the intermediate value. It is selected because it is a result that is dependent on the key and the plaintext, which should be the case for a known plaintext power analysis attack. The key is attacked as a byte at a time and the reason which makes it possible would be explained later. For each possible 256 combinations for a keybyte (called keyguesses), hypothetical power consumption values are calculated using the Hamming weight model for the writing of the corresponding byte of value $R1$. The procedure called PowerRightHalfKey in Figure 3 shows the steps to calculate the power for a given plaintext sample.

Index of the byte under attack (0 to 7 since the right half key is 64 bits) is denoted as *byteNo*. $PT1$ is the left half of the plaintext, which is a 64-bit unsigned integer represented using byte arrays due to the limitation of register width in an 8-bit microcontroller.

The *ROR* function rotates $PT1$ by 8 bits to form *temp*. Then it is added with the right half of the plaintext $PT2$ to get the value $P1$ in Figure 1. Next the *keyguess* is subjected to bitwise *xor* with the corresponding byte in the modified plaintext to get the intermediate value. This intermediate value is a single byte that is a part of value $R1$ in Figure 1. After that, the power consumption when writing this intermediate value is computed using the Hamming weight model. This computation is done for all plaintext samples used for collecting power and all keyguesses. Finally, they are compared with real power data in the power traces using Pearson correlation as discussed previously. The keyguess that returns the maximum correlation value is the correct keybyte for $K2$. By repeating this for each of eight bytes in $K2$, complete $K2$ can be recovered.

Attacking the left half key

Another set of power traces must be captured such that they include the power consumption of the second round. As the intermediate value, which depends on the round key *K*' and the plaintext, *R*2 is selected. By running the CPA again, the round key used for the second round *K*', is derived. Using the procedure called PowerRoundKey shown in Figure 3, the hypothetical power consumption for the intermediate value is calculated.

The functions *ADD*, *ROR* and *XOR* are the ones that are executed on byte arrays that represent 64-bit integers while $\oplus$ represents an *xor* operation between two 8-bit integers.

As shown in Figure 1, the values *R*1 and *S*1, which is the output of the first round is computed. In order to compute this, we require *K*2, which is the right half key that is already recovered. Then by modifying and mixing *S*1 and *R*1 as shown, the value *P*2 is computed. Then the corresponding byte in *P*2 (with respect to the byte of *K*' under attack) is subjected to *xor* with the keyguess to get the intermediate. Finally, the hypothetical power consumption for the intermediate value is found. Then as it was done for recovering *K*2, all the hypothetical power values are computed and compared with real power data using Pearson correlation to recover the round key *K*'.

However, what is required is not *K*', but *K*1. To recover *K*1 from *K*', a key reversal process must be applied to *K*' with the help of the already found *K*2 as depicted in Figure 2(b). Note that the operations here are reverse operations of the ones used to generate the round key.

K' is first subjected to *xor* with *K*2 rotated left by 3 bits. The next *xor* with value 0 (round index) does not do any change to the value, but only shown since it is in algorithm specification. When generating round keys it does change on the key except in the first round, as the round index is non-zero from the second round. Then the value at *T1* is subtracted by *K2* to get *T2*. If *T1* is less than *K2*, care must be taken. Such values for *T1* are a result of an overflow that must have occurred due to corresponding addition during key generation. In such a situation, the removed overflow bit (during addition operation at key generation step) must be added back to *T1*, and then *K2* must be subtracted from it. Finally, by left rotating *T2* by 8 bits, the left half key *K1* is computed.

When both *K*1 and *K*2 are found, the final step is to concatenate them to form the full key.

Power measurement and power analysis for Speck

Although the steps in Figure 2(a) involve taking two sets of power traces, if the number of samples in an acquisition by the oscilloscope is high enough, a single set of power traces can be taken such that power consumption for both first and the second rounds are included. However, still the analysis has to be done in two phases, due to the nature of the Speck algorithm. It is important to note that only the first two rounds of the algorithm are required for the attack, despite the total number of rounds performed.

procedure PowerRightHalfKey
$temp \leftarrow ROR(PT1, 8)$
$P1 \leftarrow ADD(temp, PT2)$
$intermediate \leftarrow P1[byteNo] \oplus keyguess$
$power \leftarrow hammingweight(intermediate)$
end procedure

procedure PowerRoundKey ▷ start of Round1
$temp \leftarrow ROR(PT1, 8)$
$P1 \leftarrow ADD(temp, PT2)$
$R1 \leftarrow XOR(P1, K2)$
$temp \leftarrow ROL(PT2, 3)$
$S1 \leftarrow XOR(temp, R1)$ ▷ start of Round2
$temp \leftarrow ROR(R1, 8)$
$P2 \leftarrow ADD(temp, S1)$
$intermediate \leftarrow P2[byteNo] \oplus keyguess$
$power \leftarrow hammingweight(intermediate)$
end procedure

Figure 3: Computing power consumption for deriving right half key and round key

Therefore, the complexity of the attack is the same, independent of the number of rounds that are performed.

Challenges faced during the attack

In this section, we visit the reasons that make an attack on Speck more challenging than the attack on AES. Further, we show how to overcome these challenges.

Larger bit widths

In AES single core software implementations, irrespective of the width of the datapath, computations are carried out byte-wise, sequentially due to the nature of the S-Box. Therefore, even if the microcontroller has a 64-bit wide datapath, still the S-Boxes are performed sequentially byte-wise. The significance of a power analysis attack when compared to a brute force attack lies in this notion. When the power consumption at a moment is dependent only on one byte, the attack can target one byte at a time and hence there are only 256 combinations for a byte. Therefore, if the key is 16 bytes, only 16 × 256 comparisons should be made.

But in a *xor* operation, if the datapath is large enough, all the bits can be calculated in parallel. The Speck implementation at Beaulieu *et al*. (2013), which is for a 128-bit key and block size variant uses 64-bit unsigned integers. The reason is that the plaintext and the key when split in half as explained previously, are 64-bit. If it is running on a microcontroller with 64-bit wide datapath, all those 64-bits would be calculated in parallel. Hence, the power consumption at the corresponding moment would be with respect to all those 64-bits. If the CPA attack is launched on 64-bits at once, it is 2^{64} combinations which are too many.

In an 8-bit microcontroller, registers are 8-bit long and hence 64-bit integers would be split into single bytes and stored separately. Therefore, the *xor* operations of the 64-bit integers also should happen sequentially, one byte after the other. The power consumed at a certain moment is only corresponding to a single byte, and, therefore, the key can be attacked byte by byte using CPA similar to what is done in AES. However, in a 16-bit microcontroller, the *xor* of 64-bit integers would be implemented using 16-bit registers and would happen sequentially, two bytes at a time. The power consumed at a certain moment is corresponding to two bytes. If CPA algorithm is launched to attack two bytes at a time, this will consume more time as there are $2^{16} \times 8$ possibilities in contrast to $2^8 \times 16$ in the former. For 32-bit and 64-bit microcontrollers, the time complexity would be further worse; $2^{32} \times 4$ and $2^{64} \times 2$, respectively. This would lead to a misconception that on a microcontroller with a wider datapath like 64 bits, a power analysis attack on Speck would be infeasible.

Since CPA algorithm has some immunity to noise (Mangard *et al*., 2007), attacking the key is still possible, byte by byte irrespective of the width of the datapath. In such cases, the power consumption for the bytes other than the byte under attack are considered noise (switching noise). To compensate the noise, more power traces are required. A mathematical perspective of how switching noise affects the number of traces is as follows. Equation 2 shows the definition of the signal to noise ratio (SNR).

$$SNR = \frac{Var(P_{signal})}{Var(P_{noise})} \quad ...(2)$$

The power consumption for the byte under attack (exploitable power) is given by P_{exp}. Noise due to the other bytes computed in parallel (switching noise) is given in P_{sw}. All other noises such as electrical noise, noise due to other components in the microcontroller is given in P_{ns}. Since P_{sw} and P_{ns} are independent random variables, equation 2 can be written as equation 3.

$$SNR = \frac{Var(P_{exp})}{Var(P_{sw}) + Var(P_{ns})} \quad ...(3)$$

As the noise in power traces increases, the value of correlation coefficients between the hypothetical and real power data decreases. This effect is given by equation 4 (Mangard *et al*., 2007).

$$\rho(H_{ck}, P_{total}) = \frac{\rho(H_{ck}, P_{exp})}{\sqrt{1 + \frac{1}{SNR}}} \quad ...(4)$$

$\rho(H_{ck},P_{exp})$ is the correlation coefficient between the Hamming weight and the exploitable power consumption. $\rho(H_{ck},P_{total})$ is the correlation coefficient with the total power consumption (including both exploitable power and noise). If the power model is good enough $\rho(H_{ck},P_{exp})$ can be approximated to be 1.

Further, the minimum number of power traces (n) approximately required to successfully derive the key in a correlation power analysis attack is given by equation 5 (Mangard *et al*., 2007).

$$n = 3 + 8\frac{z^2_{1-\alpha}}{\ln^2 \frac{1+\rho}{1-\rho}} \quad ...(5)$$

$z^2_{1-\alpha}$ is the confidence interval and ρ is same as $\rho(H_{ck}, P_{total})$ in equation 4.

Table 1: Approximate number of power traces required for different width datapaths

Datapath width (bits)	No. of bits under attack	No. of bits of noise	SNR	No. of power traces
8	8	0	0.0625	455
16	8	8	0.067	428
32	8	24	0.0556	511
64	8	56	0.0455	621

From equation 3 it is obvious that as the width of the datapath increases the switching noise increases, which in turn reduces the SNR. For example, when the datapath is 16-bits, half of the power out of the total power consumption at a moment is for the byte under attack and the other half is noise. But if the datapath is 64-bits, out of total power at a moment, 7 bytes out of 8 cause noise. As the SNR decreases, $\rho(H_{ck},P_{total})$ in equation 4 decreases. As a result according to equation 5, the number of power traces used for the attack should increase.

Let us assume a microcontroller where the power consumption for switching is proportional to the Hamming weight with a proportionality constant of k. $Var(P_{ns})$ is taken as 30 k^2 for this example. Then $Var(P_{exp})$ is found by considering the power consumption of each possible 8-bit value, which turns out to be 2 k^2. Similarly, $Var(P_{sw})$ is found by considering the power consumption for all possibilities for the corresponding bit widths. Likewise, the SNR for different width datapaths are computed using equation 3, and finally the minimum number of approximate power traces are estimated using equations 4 and 5. A confidence level of 99.99 % has been assumed and since the power model and the actual power consumption is the same in our hypothetical microcontroller, $\rho(H_{ck},P_{exp})$ is equal to 1. The computations are tabulated in Table 1.

Therefore in an ideal situation, the number of power traces does not increase by a significant amount when the width of datapath increase. But in a practical situation, the number of power traces would highly increase. The reason is that wider the datapath of a microcontroller is, more sophisticated the microcontroller becomes. Hence, $Var(P_{ns})$ which was a constant of 30 k^2 would highly increase with the presence of more components in high-end microcontrollers. Further, such high end microcontrollers are power optimised where the power consumption for a single bit swap would be small. These reasons will profoundly increase the number of required power traces in a practical situation.

Lack of confusion in the xor operation

Confusion in cryptography refers to how much the relationship between key and ciphertext is hidden. S-Box operation provides a great degree of confusion (Qu *et al*., 1999), meaning that a one-bit change in the key leads to an entirely different result. Therefore, the hypothetical power consumption values (which is the Hamming weight) for keys, which are different to one another by one bit would be entirely different. Hence, the correlation coefficient for the correct keyguess would be significantly different for keyguesses which are closer, making it possible to easily distinguish the correct key.

The *xor* operation lacks the confusion property. In the selection function $R1 = P1 \oplus K2$ used for attacking Speck, if the keyguess for $K2$ differs in one bit to the correct key $K2$, then $R1$ also changes by one bit only. Hence, the Hamming weights would be not much different for correct keyguess and the nearby key. Closer Hamming weights mean closer hypothetical power values and hence similar correlation coefficients. This small gap between the correlation coefficients of the correct key and adjacent keys make it difficult to distinguish the key. To overcome the difficulty, more power traces are required.

Identity property of xor operation

A captured power trace contains not only the power consumption during the access of the intermediate value but the nearby points as well. Therefore, if multiple keyguesses cause significant correlation at different points of the power traces, figuring out the correct key becomes a challenge. The identity property of *xor* ($X \oplus 0 = X$) in Speck introduces such issues, though the S-Boxes in AES does not.

When attacking $K2$, we use $R1$ as the intermediate value. The selection function used when calculating the hypothetical power values is $R1 = P1 \oplus K2$. The

correct keyguess for $K2$ would result in the value $R1$ and this would map to a significantly higher correlation at the moment, which $R1$ is accessed in the power trace. However, note that $P1 = P1 \oplus 0$ also holds true. Therefore, if the power trace contains the access of $P1$ that point would also cause a significantly higher correlation with the keyguess 0×00. Since reading $P1$ is happening very close to the writing of value $R1$ (Figure 1), the power trace would have both the points making it difficult to figure out the exact key.

One obvious solution is selecting the non-zero value as the key. But that is not always possible especially in microcontrollers, where reading consumes more power than writing. If the plaintext reading consumed more power than writing the result, keys that are one bit different to the 0×00 key also might cause high correlations, due to lack of confusion in *xor* operation that was already discussed. Therefore, to avoid this situation, it requires to trim power traces, so that the part of the traces that includes reading of $P1$ is removed. In any unprotected cryptosystem, the clock period is always fixed and the implementation of a cryptosystem is assumed to be known to anybody. Hence, the information about its instructions is known. Therefore, the period from the trigger to the end of the instruction that reads $P1$ can be computed, and the respective samples in the power trace can be removed.

In AES, such problems of falsely correlated keys do not exist as S-Box lookups are nonlinear operations with no identity property. But when the only viable selection function is the *xor*, the trimming of power traces would be required.

Issue caused due to the complement

In microcontrollers with pre-cleared buses (all bits on the bus are initially set to zero before writing a value on the bus), the number of bits swaps when reading or writing is equal to the Hamming weight making it an ideal power model. But if it is a pre-charged bus (bits are initially set to one), then the hypothetical power must be computed by subtracting the Hamming weight by the total number of bits on the bus.

For attacking AES, knowing the type of the bus is not a strict requirement. Whatever the bus type is, Hamming weight can always be used for finding hypothetical power values. If the bus is pre-cleared, the returned correlation coefficient for the correct key would be a positive value while for a pre-charged bus it would be negative. Hence taking the absolute value of the correlation coefficients will return the correct key despite the type of the bus for AES.

But this is not the case for Speck due to the property of *xor* in which, if $R1 = P1 \oplus K2$, then $R1' = P1 \oplus K2'$, where $R1'$ is the complement of $R1$. Therefore, in a pre-cleared bus, when the correct keyguess $K2$ causes the highest positive correlation coefficient, $K2'$ will cause a correlation coefficient of the same magnitude but negative. For a pre-charged bus, it would be *vice-versa*. Hence, if the absolute value of the correlation coefficients is used, both $K2$ and $K2'$ are returned as the best matches making it impossible to select between the two. In contrast, the S-Box operation in AES $R = SBOX(P,K)$ does not mean $R' = SBOX(P,K')$, making it possible to use the absolute value despite the type of the bus.

Hence, for attacking Speck, the knowledge on the type of the bus would simplify the attack. Otherwise, for a 128-bit key, two possibilities per each keybyte would return 2^{16} possible keys, where the attacker would have to run a brute force among them to find the exact one.

RESULTS AND DISCUSSION

In this section we report the evaluation of the vulnerability of Speck for power analysis attack experimentally, by following the procedures explained previously.

Experimental setup

We attempt the attack on Speck algorithm running on PIC microcontrollers by Microchip.

We have an 8-bit cryptosystem and two 16-bit cryptosystems. We performed the attack on both 8-bit and 16-bit microcontrollers to show that Speck is breakable despite the width of the datapath as discussed in the Methodology. Cryptosystem-1 is the 8-bit one based on a 18F2550 microcontroller. Cryptosystem-2 and 3 are 16-bit ones based on 24HJ32GP302 and 24FJ32GA002 microcontrollers, respectively. The microcontroller on cryptosysem-2 is designed for high performance while cryptosystem-3 has one optimised for low power. This was used to demonstrate that the number of required traces for a successful attack not only depends on the algorithm, but also on numerous other factors such as power characteristics of the architecture and noise.

Cryptosystems communicate with the computer *via* USB, using a USB to RS232 TTL FTDI module. The cryptosystems sit in an infinite loop to accept plaintext from the computer and does encryption. The variant of

Speck with 128-bit block size and 128-bit key has been implemented using CCS PIC C. As the compiler for 8-bit microcontrollers did not support 64-bit unsigned integers, they were implemented using unsigned char arrays. However, for the 16-bit microcontrollers, inbuilt 64-bit integers in CCS PIC C were used as the compiler supported them. For the purpose of measuring power, a 100-ohm resistor has been connected serially to the microcontroller along the power line (Gamaarachchi *et al*., 2015). Power measurements were taken by using a Tektronix MSO2012B digital oscilloscope connected *via* a USB to a computer with Matlab installed. The oscilloscope has a sampling rate of 1 Giga-samples per second and a bandwidth of 100 MHz. A pin of the microcontroller has been programmed to provide the trigger needed for the oscilloscope. Since CPA algorithm is more than 1000 times faster on a GPU compared to a CPU (Gamaarachchi *et al*., 2014) the analysis was done on a computer that consists of an NVIDIA Tesla C2075 GPU.

Results for 8-bit cryptosystem (cryptosystem-1)

A set of 500 randomly generated plaintext samples were fed, and power traces were obtained. As explained in the Methodology two sets of 500 power traces were obtained separately for round 1 and round 2 of the encryption to attack the right half key and the left half key, respectively.

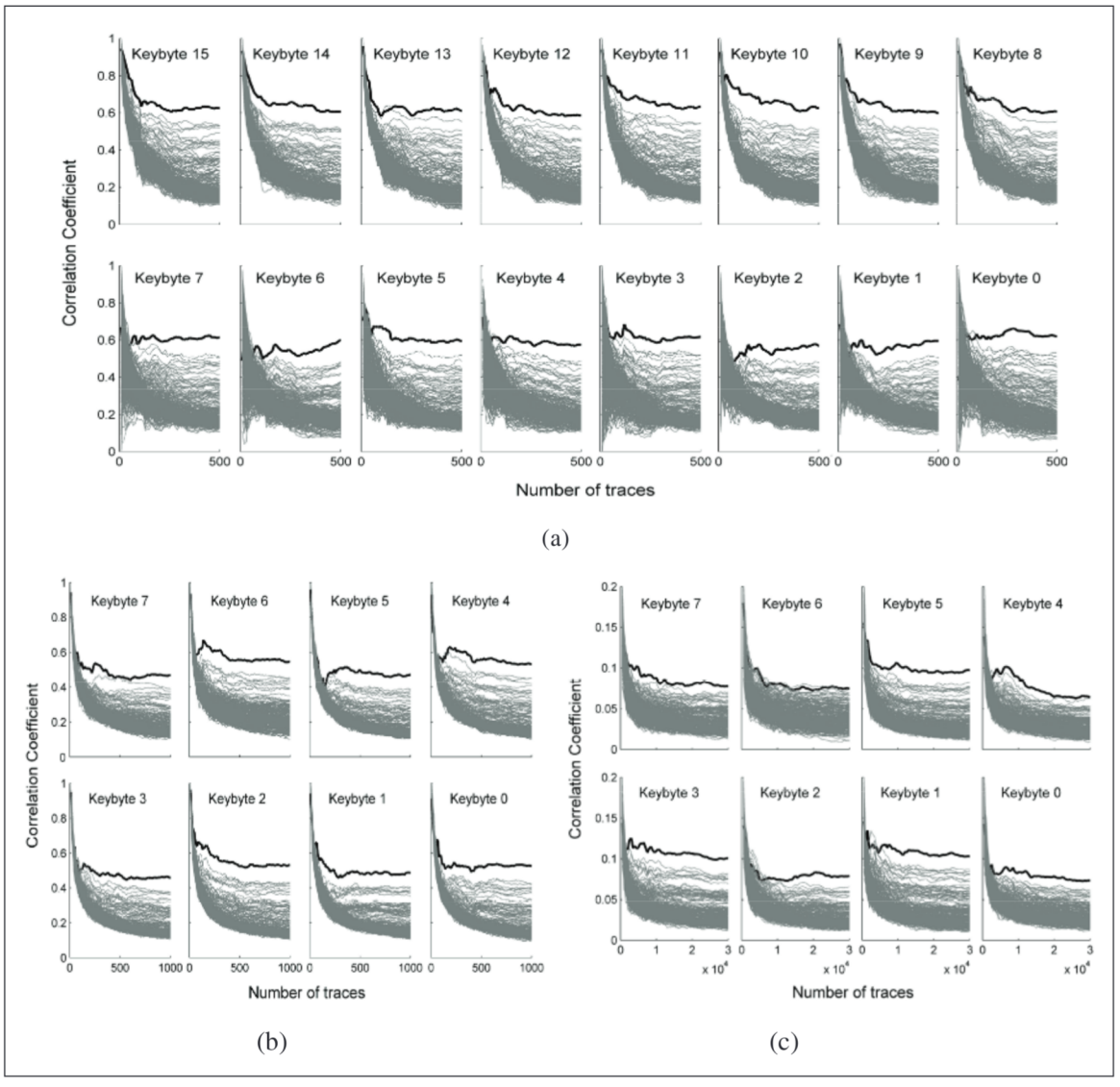

Figure 4: Variation of correlation coefficient with number of power traces for the (a) 8-bit cryptosystem; (b) high power 16-bit cryptosystem; (c) low power 16-bit cryptosystem

As presented earlier, using the traces without trimming led the key being falsely returned as zero. However, when traces were trimmed, the correct key could be derived. Figure 4(a) shows how the correlation coefficient changes with the number of power traces for each keybyte. In each of the 16 graphs, there are 256 lines where a single line represents one of 256 keyguesses for that respective keybyte. The correct keyguess is shown in black, while the others are shown in grey.

The x-axis of each graph denotes the number of power traces used for the attack, and the y-axis indicates the peak correlation coefficient observed for that keyguess. It can be observed that for all the keybytes, the dark line representing the correct key lies above all other lines after about 300 power traces, indicating the highest correlation. At this stage, the correlation coefficient of the right key is larger than that of the second correlated key by a recognisable amount to be correctly identified.

The attack was attempted multiple times with multiple keys with 500 power traces captured for each, where the full key could be successfully recovered in all cases. Hence, the global success rate (Department of the Télécom ParisTech French University, 2016) is evaluated to be 100 % for 500 power traces. Table 2 shows the time required when attacking using 500 power traces, in which the total was about half an hour.

Results for 16-bit cryptosystems (cryptosystem-2 and 3)

For cryptosystem-2 made of the high-performance 16-bit microcontroller, power traces for 1000 randomly generated plaintext samples were obtained. Figure 4(b) shows the plots for this 16-bit cryptosystem. Only the first 8 keybytes of 16 are shown because the rest are similar. After about 800 traces, the correct key lies above the other keys at an adequate level to be recognised. The increase in this number for the 16-bit system when compared to the 8-bit system is the noise introduced by other bytes computed in parallel and the additional components in the 16-bit microcontroller, as discussed previously. The time required to recover the full key under the two phases was around one hour.

Table 2: Time taken for an attack on 8-bit Speck cryptosystem

Step	Time taken / s
Phase 1: Collecting power traces	913.52
Phase 1: Running CPA	28.97
Phase 2: Collecting power traces	907.63
Phase 2: Running CPA	28.63
Sum	1878.75

The respective plots for an attack on cryptosystem-3 made of a low power microcontroller are shown in Figure 4(c). The number of power traces required is much higher when compared to the previous high performance 16-bit microcontroller. The correct key for keybyte 6 and keybyte 4 have begun to appear in 30000 traces, but yet the gap from the second correlated key is still very small to be recognised as the correct key. However, other keybytes by about 20000 traces show a recognisable deviation in the correct key although the number is again comparatively larger. The fact that the microcontroller is power optimised, contributes the increase in the number of power traces. Hence, the number of power traces highly depends on the features of the microcontroller as well, rather than the datapath width alone.

Analysis of data leakage points

Figure 5(a) depicts how the correlation coefficient varies with time for the correct keyguess in a certain keybyte. The value of this particular keybyte was set to 0×07 where 3 of 4 bits are 1. The time is represented using the sample number in the acquired power traces. The time window includes operations starting from reading of $R1$ to the writing of $S1$ in Figure 1. Two correlation peaks are clearly visible. The first one corresponds to writing of the considered intermediate value $R1$ to memory after the *xor* between $P1$ and $K2$. The second peak corresponds to reading of $R1$ back from the memory, to be used for the *xor* that generates $S1$. The second correlation peak is larger than the other. While the second peak has exceeded 0.3 correlation, the first peak is lesser than that. The reason can be that the power consumption for reading from memory is larger than that for writing in the used microcontroller. Also if a peak is zoomed (for second peak visible even without zooming) it is observable that there are a few more closely packed peaks. These are due to transfer of the intermediate value in between registers. Figure 5(b) shows the same graph now for the keyguess 0×00. There is a single peak in the graph, which is corresponding to reading of $P1$ from memory. As explained previously this false correlation occurs due to the identity property of *xor*. This peak lies just left of the first peak in Figure 5(a) as the reading of $P1$ happens just before writing $R1$, but as they happen one after the other it is not visible in the graphs. Note that the single peak in Figure 5(b) corresponding to zero key has a correlation value larger than the correlation peaks in Figure 5(a). Hence, unless the power traces are trimmed zero will be returned as the most correlated key.

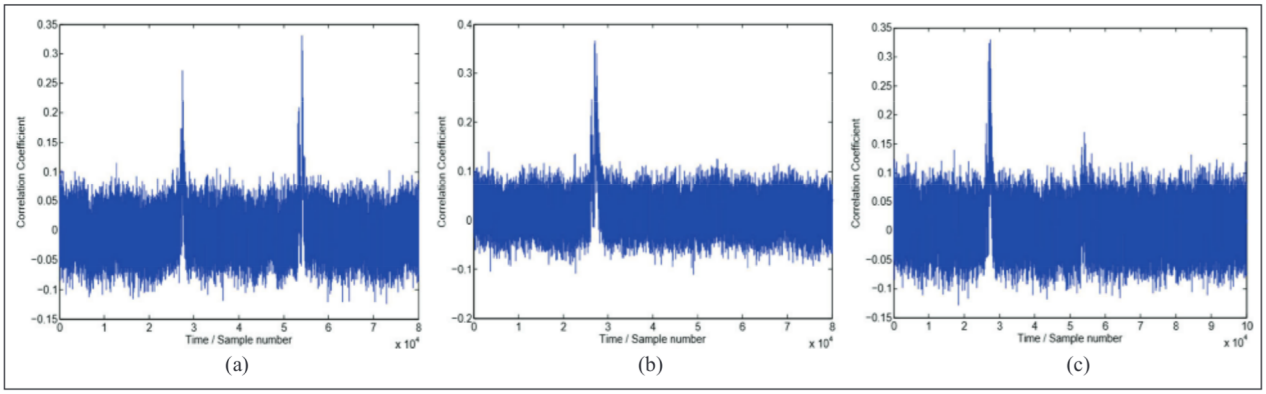

Figure 5: Variation of correlation coefficient with time for (a) correct keyguess 0 × 07; (b) keyguess 0 × 00; (c) keyguess 0 × 01

Figure 5(c) shows the graph for the keyguess 0 × 01. This is different to keyguess 0 × 00 by only 1 bit. Note that, it also has two correlation peaks. The first significant correlation peaks are due to the reading of $P1$ from memory, which is the same reason that caused the peak in Figure 5(b). Since 0 × 01 is only different by 1 bit to 0 × 00 the power consumption difference is small enough, and hence cause a significant correlation. Note that this peak is having a very similar value to the largest peak in Figure 5(a). Therefore, there can be situations where even after ignoring the zero key, the next highest correlated key is a wrong one rather than the correct one. That is why it was stated earlier that trimming the power traces such that the part where $P1$ read is recommended. Also, note that Figure 5(c) contains the second peak as well which is not that large but significant enough to be recognised. This corresponds to the location where $R1$ is read from memory to be used for the *xor* that generated value $S1$, which is corresponding the same reason that caused the second peak in Figure 5(a). Although 0 × 01 is two bits different to the correct key 0 × 07 still it causes a notable correlation.

CONCLUSION

It is shown that a recent addition to the block ciphers known as Speck, is vulnerable to correlation power analysis attack. The experimental setup demonstrates that a Speck cryptosystem implemented on different microcontrollers can be broken in less than an hour. Given the lightweight nature of Speck, it has the potential of becoming the block cipher of the future where the embedded devices are going to be ubiquitous. Therefore, we emphasise the importance of providing countermeasures against the same.

REFERENCES

1. Abed F., List E., Lucks S. & Wenzel J. (2014). Differential cryptanalysis of round-reduced simon and speck. *International Workshop on Fast Software Encryption*, London, UK, 3 – 5 March, pp. 525 – 545.
2. Beaulieu R., Shors D., Smith J., Treatman-Clark S., Weeks B. & Wingers L. (2015). Simon and speck: block ciphers for the internet of things. *IACR Cryptology ePrint Archive* **2015**: 585.
3. Beaulieu R., Treatman-Clark S., Shors D., Weeks B., Smith J. & Wingers L. (2013). The simon and speck families of lightweight block ciphers, *Proceedings of the 52nd Annual Design Automation Conference (DAC '15),* San Francisco, USA, 8 - 12 June, pp. 1 – 6.
4. Biryukov A., Roy A. & Velichkov V. (2014). Differential analysis of block ciphers simon and speck. *Fast Software Encryption* (eds. C. Cid & C. Rechberger), volume 8540, pp. 546 – 570. Springer, Heidelberg, Germany.
5. Brier E., Clavier C. & Olivier F. (2004). Correlation power analysis with a leakage model. *Cryptographic Hardware and Embedded Systems* (eds. M. Joye & J.J. Quisquater), volume 3156, pp. 16 – 29. Springer, Heidelberg, Germany. DOI: https://doi.org/10.1007/978-3-540-28632-5_2
6. Clavier C., Coron J.S. & Dabbous N. (2000). Differential power analysis in the presence of hardware countermeasures. *International Workshop on Cryptographic Hardware and Embedded Systems - CHES 2000*, Springer, Heidelberg, Germany, pp. 252 – 263. DOI: https://doi.org/10.1007/3-540-44499-8_20
7. Department of the Télécom ParisTech French University (2016). *DPA Contest v2*, Available at *http://www.dpacontest.org/v2* , Accessed 21 February 2016.
8. Dinur I. (2014). Improved differential cryptanalysis of round-reduced speck. *International Workshop on Selected Areas in Cryptography*, Springer, Cham, Switzerland, pp. 147 – 164. DOI: https://doi.org/10.1007/978-3-319-13051-4_9

9. Eisenbarth T., Kasper T., Moradi A., Paar C., Salmasizadeh M. & Shalmani M.T.M. (2008). On the power of power analysis in the real world: a complete break of the keeloq code hopping scheme. *Crypto* **2008**(5157): 203 – 220.
10. Gamaarachchi H., Ganegoda H. & Ragel R. (2015). The a to z of building a testbed for power analysis attacks. *Proceedings of the 2015 IEEE 10th International Conference on Industrial and Information Systems (ICIIS)*, Peradeniya, Sri Lanka, 18 – 20 December, pp. 501 – 506. DOI: https://doi.org/10.1109/ICIINFS.2015.7399063
11. Gamaarachchi H., Ragel R. & Jayasinghe D. (2014). Accelerating correlation power analysis using graphics processing units (GPUs). *Proceedings of the 2014 7th International Conference on Information and Automation for Sustainability (ICIAfS)*, Colombo, Sri Lanka, 22 – 24 December, pp. 1 – 6. DOI: https://doi.org/10.1109/ICIAFS.2014.7069547
12. Kocher P., Jaffe J. & Jun B. (1999). Differential power analysis. *Advances in Cryptology CRYPTO99*, pp. 388 – 397. Springer, Heidelberg, Germany. DOI: https://doi.org/10.1007/3-540-48405-1_25
13. Liu J., Yu Y., Standaert F.X., Guo Z., Gu D., Sun W., Ge Y. & Xie X. (2015). Small tweaks do not help: differential power analysis of MILENAGE implementations in 3g/4g USIM cards. *European Symposium on Research in Computer Security,* Springer, Cham, Switzerland, volume 9326, pp. 468 – 480. DOI: https://doi.org/10.1007/978-3-319-24174-6_24
14. Liu Y., Fu K., Wang W., Sun L. & Wang M. (2016). Linear cryptanalysis of reduced-round SPECK. *Information Processing Letters* **116**(3): 259 – 266. DOI: https://doi.org/10.1016/j.ipl.2015.11.005
15. Mangard S., Oswald E. & Popp T. (2007). *Power Analysis Attacks: Revealing the Secrets of Smart Cards (Advances in Information Security)*, pp. 1 – 165, Springer-Verlag New York, Inc., Secaucus, USA.
16. Martinasek Z., Clupek V. & Krisztina T. (2013). General scheme of differential power analysis, *2013 36th International Conference on Telecommunications and Signal Processing (TSP),* Rome, Italy, 2 – 4 July, pp. 358 – 362. DOI: https://doi.org/10.1109/TSP.2013.6613952
17. Messerges T.S., Dabbish E.A. & Sloan R.H. (1999). Investigations of power analysis attacks on smartcards. *Smartcard* **99**: 151 – 161.
18. Petrvalsky M., Drutarovsky M. & Varchola M. (2013). Differential power analysis of advanced encryption standard on accelerated 8051 processor. *23rd International Conference Radioelektronika*, Pardubice, Czech Republic, 16 – 17 April, pp. 334 – 339. DOI: https://doi.org/10.1109/RadioElek.2013.6530942
19. Petrvalsky M., Drutarovsky M. & Varchola M. (2014). Differential power analysis attack on ARM based AES implementation without explicit synchronization. *24th International Conference Radioelektronika*, Bratislava, Slovakia, 15 – 16 April, pp. 1 – 4. DOI: https://doi.org/10.1109/Radioelek.2014.6828434
20. Qu C., Seberry J. & Pieprzyk J. (1999). On the symmetric property of homogeneous boolean functions. *Information Security and Privacy*, pp. 26 – 35. Springer, Heidelberg, Germany. DOI: https://doi.org/10.1007/3-540-48970-3_3
21. Rambus Corporation (2016). *MStar to Use Cryptography Research DPA Countermeasures to Ward Off Attacks in SetTop Box Solutions.* Available at *http://www.rambus.com/mstar-to-use-cryptography-research-dpa-countermeasures-to-ward-off-attacks-in-set-top*, Accessed 15 February 2016.
22. Speck cipher, Available at *https://en.wikipedia.org/wiki/Speck_(cipher)*, Accessed 15 February 2016.
23. Tupsamudre H., Bisht S. & Mukhopadhyay D. (2014). Differential fault analysis on the families of SIMON and SPECK ciphers. *Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, Busan, South Korea, 23 September, pp. 40 – 48. DOI: https://doi.org/10.1109/FDTC.2014.14
24. Zhu N., Zhou Y. & Liu H. (2013). Counteracting leakage power analysis attack using random ring oscillators. *International Conference on Sensor Network Security Technology and Privacy Communication System (SNS & PCS)*, Nangang, China, 18 – 19 May, pp. 74 – 77.