Performance Evaluation of Reconfigurable Lightweight Block Ciphers

Mostafa Hashempour Koshki
Faculty of Electrical and Computer Engineering
University of Tabriz
Tabriz, Iran
email: m.hashempour71@gmail.com

Reza Abdolee
Department of Computer Science
California State University, Channel Islands (CSUCI)
Camarillo, CA 93012, USA
email: reza.abdolee@csuci.edu

Behzad Mozaffari Tazekand
Faculty of Electrical and Computer Engineering
University of Tabriz
Tabriz, Iran
email: mozaffary@tabrizu.ac.ir

***Abstract*—The growing number of connected devices and massive data in the Internet of Things (IoT) cause information to encounter different types of attacks. One solution to the security problem of computationally intensive traditional cryptographic algorithms for IoT environments is stronger lightweight cryptography. This paper evaluates the security performance of reconfigurable lightweight block ciphers featuring round order and internal parameter randomization. We evaluate these lightweight block ciphers and compare their performance with that of conventional lightweight block ciphers in terms of execution time, energy consumption and throughput. The simulation results of reconfigured-based lightweight block ciphers show significant improvement in security performance with minor and negligible changes in energy-throughput performances.**

***Index Terms*—Security performance, Cybersecurity, Reconfigurable lightweight block ciphers, Round order randomization, Internet of Things (IoT).**

I. Introduction

Traditional cryptography algorithms have been designed for a network of computers with high processing power [1]. These algorithms are not suitable for communication devices with low computational power or storage spaces [2]. Alternatively, a new cryptography method called Lightweight Cryptography as a subset of cryptography was introduced [3]. A strong lightweight cryptography algorithm provides appropriate security level for resource-constrained devices. There are several categories in lightweight cryptography, including Lightweight Block Cipher (LWBC), Lightweight Stream Cipher (LWSC), and Lightweight Hash Function (LWHF) [4]. Block Ciphers (BC) are used in resource-constrained end-devices because they support keys and messages in varying sizes that can be adaptively changed. Lightweight block ciphers are defined based on the key size, block length, number of rounds, and key schedule structure that are smaller and simpler than conventional block ciphers.

Several types of LWBC have been proposed for IoT systems [5]. Some of the proposed LWBCs for resource-constrained devices have been derived from the optimization of conventional block ciphers, such as Welch-Gong cryptography (WG-8) [6] or Lighter algorithms from Advanced Encryption Standard (AES), Rivest Cipher-5 (RC-5), eXtended Tiny Encryption Algorithm (XTEA), and Data Encryption Standard (DES) [7]–[10]. In LWBC algorithms, smaller block/key sizes are considered for faster processing and consuming less resources. The PRESENT algorithm is provided with an 80-bit key [11] and the TWINE algorithm is presented with both 80-bit and 128-bit keys [12]. The number of rounds in this type of cryptography should be limited to save time to encrypt/decrypt the message. The Hummingbird [13] only has 4 rounds. A simple key schedule is used to produce subkeys such as converting a 128-bit key to 32-bit subkeys in the TEA block cipher by using a simple procedure [14]. Several LWBC algorithms are introduced as Generalized lightweight Feistel Network (GFN) categories for IoT systems such as PICCOLO, TWINE, and CLEFIA, with a trade-off between security and being lightweight [15] [16].

An attack on LWBC algorithms becomes more complicated by adding reconfigurability features and randomizing key parameters in their structures [17]. One way to do this is to use a Random Number Generator (RNG) to improve the security of the algorithm in exchange for small changes in the computational complexity and the consumption of resources and memory.

In this paper, we investigate the security performance of PICCOLO, TWINE, and PRESENT lightweight block ciphers via the round order randomization concept by using a pseudo-random number generator. In the Reconfigurable Hardware (RCH) method [18], only the order of the round keys in algorithms was reconfigured. However, in our proposed method, not only the order of the rounds is randomized, but also some internal parameters are randomized in key scheduling and data processing parts. For example, in PICCOLO algorithms, besides the order of round keys, the constant values of each round are also reconfigured separately. In fact, the initial table of constant values is shuffled. Additionally, in the 128-bit master key, which is initially divided into 8 subkeys (16-bit), randomization has been implemented. In this way, these subkeys are randomly selected to produce whitening keys and round keys. Even the number of rounds is reconfigured and randomly selected (11 to 41 for PICCOLO128 and 11 to 30 for PICCOLO80). In the key scheduling section of the TWINE algorithm, besides the order of rounds, the constant values of each round and the S-box are reconfigured separately and are randomly assigned to each round. In the data processing section, the table of $\pi[j]$ (Block indexes) values is randomized. In PRESENT algorithms, randomization of round orders and internal values such as S-box layer and P-layer is performed. In addition, the 64-bit algorithm key is randomly selected from the 80 or 128-bit master keys. In this algorithm, like PICCOLO, the number of rounds is considered variable between (20 to 35) for both key sizes of the PRESENT algorithm. We finally evaluate these round order and internal parameters of reconfigurable-based lightweight block ciphers and compare their performance with that of conventional lightweight block ciphers in terms of execution time, energy consumption, and throughput.

The paper is organized as follows. In Section II, we describe the proposed reconfigurable algorithms. Section III presents the performance analysis of LWBCs using round order and internal parameters randomization. Cryptanalysis of the randomized LWBCs is presented in Section IV. Finally, we conclude the paper in Section V.

II. Reconfigurable LWBC Algorithms

Using a RNG, the number of rounds of the algorithm, key scheduling part, and order of production of the round keys can be randomized. In this way, the key schedule becomes a pseudo-random function. If the algorithm is run successively for specific key and plaintext, different ciphers are produced. Therefore, a chosen-plaintext attack on the algorithm becomes impossible.

The steps of the randomization process are as follows. At first, a suitable range for the algorithm is defined and then by using an RNG, the number of rounds in the algorithm is determined. The randomization is also implemented in the key scheduling part with the same value generated by the RNG.

In this research, we introduce new reconfigurable-based algorithms for PICCOLO, TWINE, and PRESENT algorithms. They are, respectively, presented as Algorithms 1,2, and 3 in this paper. Algorithm 1 features different number of rounds and different order of round keys. In addition, it has a different key-table due to the randomizing order of master keys for the key scheduling part in each implementation. The RNG generates 30 and 41 random numbers for PICCOLO80 and PICCOLO128, respectively, that are indicated by $Random$ parameter in the algorithm. Therefore, the value of $fullround$ of the algorithm is the same as $Random$ parameter (30 and 41 for PICCOLO80 and PICCOLO128, respectively). In PICCOLO, each round has a constant value (con) that is computed by the number of each round. Roundkeys are reconfigured based on k and con, which are reconfigured based on n and

Algorithm 1: Proposed Reconfigurable PICCOLO128

```
Data: The master key $k_j$ $0 \leq j < n$, $n = 8$,
  $fullround = 41$
Reconf $(k_j, n)$ ◁ Reconfigure $k$ based on $n$
Reconf $(con_i, fullround)$ ◁ Reconfigure $con$ based on
  $fullround$
Reconf $(Roundkeys, k_j.con_i)$ ◁ Reconfigure $Roundkeys$
  based on $con$ and $n$
$i \leftarrow 0$
for $0 \leq i < fullround$ ◁ Store Random Index do
    $Random \leftarrow RNG(seed)$
    while $i < n$ do
        $rndi \leftarrow Random$ mod $n$
        $knew[i] \leftarrow k[rndi]$ ◁ Reconfigure $k$ based on $n$
    end
    $r \leftarrow Random$ mod $fullround$
    $newcon(2i, 2i+1) \leftarrow con(2r, 2r+1)$
    if i=n+1 ◁ Reconfigure number of rounds then
        $rndround \leftarrow r$
        if $rndround < 10$ then
            $rndround \leftarrow rndround + 13$
            if $rndround\ mod\ 2 = 0$ then
                $rndround \leftarrow rndround + 1$
            end
        end
    end
end
$Roundkeys \leftarrow keyschedule[con_i, knew]$
$G_r = Enc.Roundkeys_r$
```

$fullround$, respectively. In addition, the whole algorithm is randomized based on the number of rounds when the value of i is equal to $n+1$. The number of rounds, order of round-keys, and key-table of the key scheduling part were reconfigured as shown in Algorithm 1. In this way, a temporary array is produced to store the random number 0 through $fullround$ where $fullround$ is the maximum value of PICCOLO rounds number. If reconfigurable hardware is used to control random values, randomization has minimal effect on the performance of algorithms. The memory requirements will be reduced when the hardware RNG creates random values. By using $rndi$, the order of key masters will be randomized and stored in a new table which is indicated by $knew$. The generated n^{th} random number is used to set the number of rounds in module $fullround$. Finally, we encrypt in r rounds and with a new table of master keys for the key scheduling section.

In the randomization of TWINE, both data processing and key scheduling sections have been reconfigured by randomized order of S-box as their internal elements. The order of a permutation of block indexes, which are indicated by $\pi[j]$ and $\pi^{-1}[j]$ for encryption and decryption in the data processing part, are randomized. In the key scheduling part, the order of constant value and order of round keys, along with S-box, are reconfigured. Because of the randomization of $\pi[j]$, the order of $\pi^{-1}[j]$ has changed. The RNG generates 36 random numbers for both key sizes of algorithms. The order of S-box, $\pi[j]$ and con_i were computed based on the number of rounds in the randomization of the TWINE algorithm. The original key is divided into 5 and 8 16-bit keys for TWINE80 and

Algorithm 2: Proposed Reconfigurable TWINE128

```
Data: The master key $k_j$ $0 \leq j < n$, $n = 8$,
  $NbRound = 36$
Reconf $(k_j, n)$ ◁ Reconfigure $k$ based on $n$
Reconf $(con_i, NbRound)$ ◁ Reconfigure $con$ based on
  $NbRound$
Reconf $((\pi[j] \ \& \ \pi^{-1}[j], 16))$ ◁ Reconfigure $\pi[j]$ and
  $\pi^{-1}[j]$ based on 16
Reconf $(S(x), 16)$ ◁ Reconfigure $S$ based on 16
Reconf $(Roundkeys, k_j.con_i.S(x))$
$i \leftarrow 0$
for $0 \leq i < NbRound$ ◁ Store Random Index do
    $Random \leftarrow RNG(seed)$
    while $i < n$ do
        $rndi \leftarrow Random$ mod $n$
        $knew[i] \leftarrow k[rndi]$ ◁ Reconfigure $k$ based on $n$
    end
    $r \leftarrow Random$ mod $NbRound$
    $newcon(i) \leftarrow con(r)$
    while $i < 16$ ◁ Reconfigure number of rounds do
        $rndi \leftarrow Random$ mod 16
        $Snew[i] \leftarrow S[rndi]$
        $\pi[i] \leftarrow \pi[rndi]$
        $\pi^{-1}[\pi[i]] \leftarrow i$
    end
end
$Roundkeys \leftarrow keyschedule[newcon_i, knew, S_i]$
$Enc \leftarrow Encryption[S_i, \pi[i]]$
$Dec \leftarrow Decryption[S_i, \pi^{-1}[i]]$
$G_r = Enc.Roundkeys_r$
$G_r^{-1} = Dec.Roundkeys_r$
```

TWINE128, respectively. As seen, an RNG is seeded with a primary value in the randomizing path. A temporary array is generated to store the random number 0 through $NbRound$ which is 36 in both the TWINE80 and TWINE128 algorithms. By randomizing the con, k, and S, the order of round keys has changed. The entire encryption and decryption process has been reconfigured by randomizing the S_i, $\pi[j]$, and $\pi^{-1}[j]$.

Finally, for the PRESENT, some parameters have been considered for reconfiguring the algorithm. Like PICCOLO algorithm, PRESENT is reconfigured by randomizing the number of rounds (between 20 and 32). The S-box and P-layer have also been randomized to reconfigure both the key scheduling and data processing parts of the PRESENT algorithm. Another change is in the selection of a 64-bit algorithm key from an 80-bit or 128-bit master-key. In the original cipher, the most significant 64 bits of the master-key are selected as the algorithm key, while in the proposed method, the mentioned key is obtained randomly from the 80-bit or 128-bit key. The order of the round keys is also randomized. The RNG generates 64 random numbers for the PRESENT algorithm with both of the key sizes (80 and 128). Since S-box, P-layer, and master key are reconfigured, the order of round keys is reconfigured based on them and $NbRound$. The computations of the parameters which are involved in the algorithm randomization are indicated in Algorithm 3.

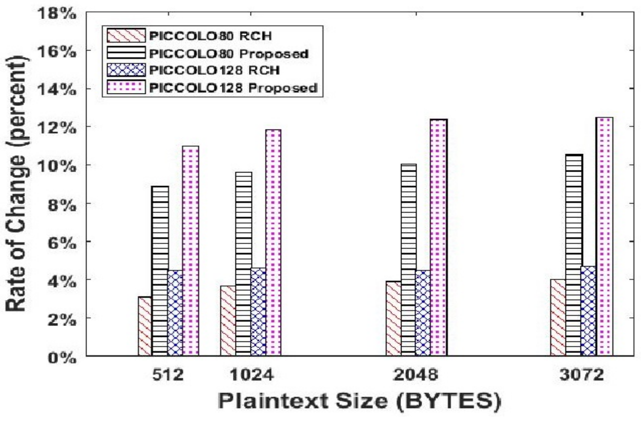
In our implementation, we use the STM32F401RE, which is an STM32 (ARM Cortex M4) microcontroller. Its ARMv7E-M architecture with 3 stage pipelining results in ideal average Clocks Per Instruction (CPI) of 1.67 [19]. The performance metrics parameters such as energy consumption, execution time, memory efficiency, and throughput of the proposed algorithms were analyzed for PICCOLO, TWINE, and PRESENT with a pseudo-random number generator. For randomized PICCOLO, the number of rounds, order of subkeys, and constant value of each round are reconfigured. The possible number of rounds is 17 and 28 for PICCOLO80 and PICCOLO128, respectively. In other words, the total possible combination for PICCOLO80 and PICCOLO128, respectively, is $17 \times r! \times con!$ and $28 \times r! \times con!$ where $r!$ is the permutations of the subkeys and $con!$ is the permutations of each round's constant values. The order of internal keys, S-boxes, diffusion of round indexes, and constant value of rounds are randomized in TWINE algorithms implementation. The permutation of the internal keys with $n!$ (n=5 for TWINE80 and n=8 for TWINE128), $con!$ with $NbRound!$, S and $\pi[j]$ with 16! has made the entire possible combination of both TWINE algorithms, so the permutation of algorithms is $n! \times NbRound! \times 16!$. The proposed PRESENT algorithms have the same permutation for both key sizes. The number of rounds, algorithm key, order of round keys, S-box, and P-layer have 12, 64!, 32!, 16!, and 64! possibilities. Therefore, the total permutation of the PRESENT algorithms is $12 \times 64! \times 32! \times 16! \times 64!$.

Algorithm 3: Proposed Reconfigurable PRESENT128

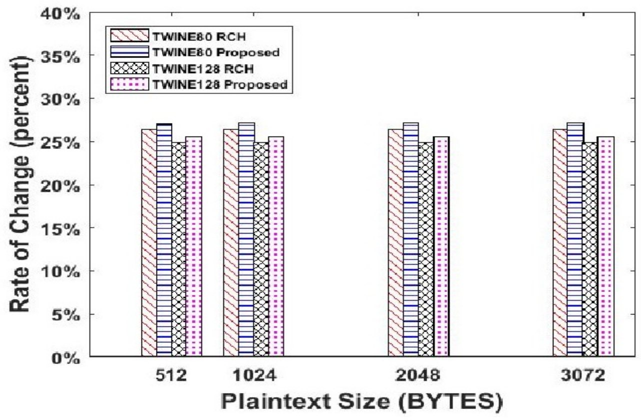
```
Data: The master key $K_1 28$, Round Key $rk$ Bits of
  algorithm's key $k_i$ $0 \leq i < ksize$, $ksize = 64$, Maximum
  of $NbRound$ is 32
Reconf $(k_i, ksize)$ ◁ Reconfigure $k$ based on $ksize$
Reconf $(rk, NbRound, k_i, Sbox, Player)$ ◁ Reconfigure
  $rk$ based on $NbRound$, $k_i$, $Sbox$, and $Player$
Reconf $(Sbox, 16)$ ◁ Reconfigure $Sbox$ based on 16
Reconf $(Player, 64)$ ◁ Reconfigure $Player$ based on 64
$i \leftarrow 0$
for $0 \leq i < 64$ ◁ Store Random Index do
    $Random \leftarrow RNG(seed)$
    if i=0 ◁ Reconfigure number of rounds then
        $NbRound \leftarrow Random$ mod 12 + 20
    end
    $r \leftarrow Random$ mod 64
    $Player[i] \leftarrow Player[r]$
    while $i < NbRound$ do
        $r \leftarrow Random$ mod $NbRound$
        $rk[i] \leftarrow rk[r]$
    end
    while $i < 16$ do
        $r \leftarrow Random$ mod 16
        $Sbox[i] \leftarrow Sbox[r]$
    end
end
$rk \leftarrow keyschedule[k_i, NbRound, Sbox, Player]$
$G_r = Enc.rk$
```

III. Performance Analysis

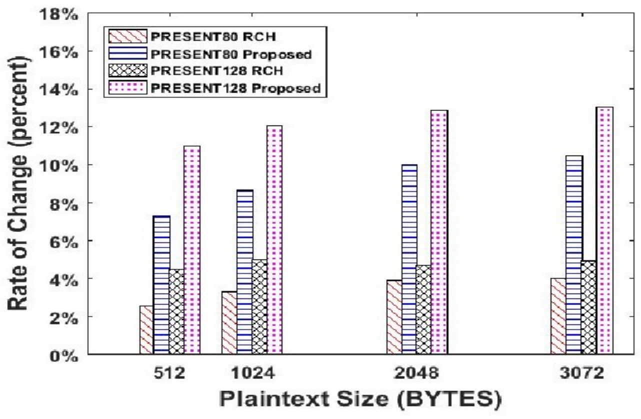
In our analysis, we investigate execution time, memory consumption, energy consumption, and throughput for PICCOLO, TWINE, and PRESENT.



(a)

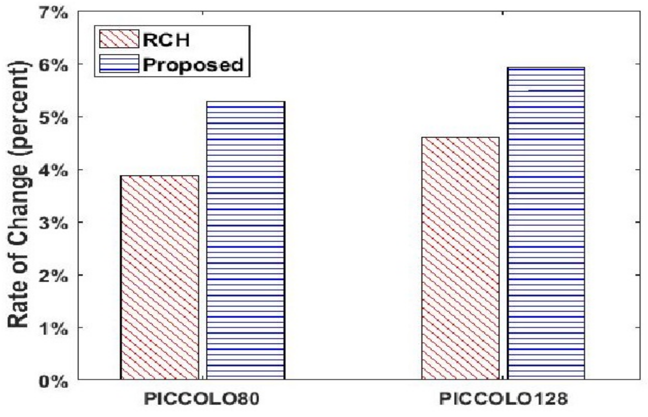


(b)

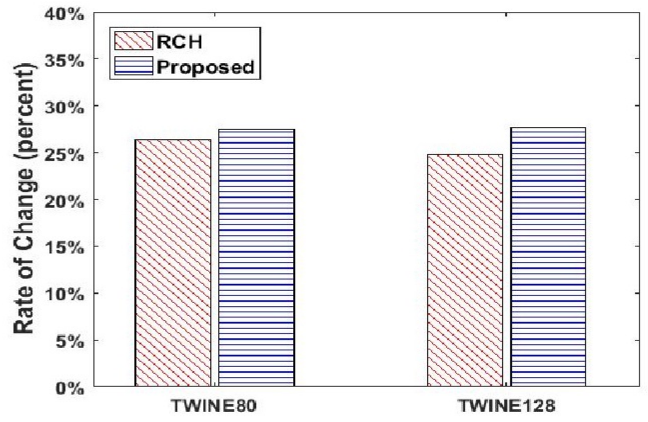


(c)

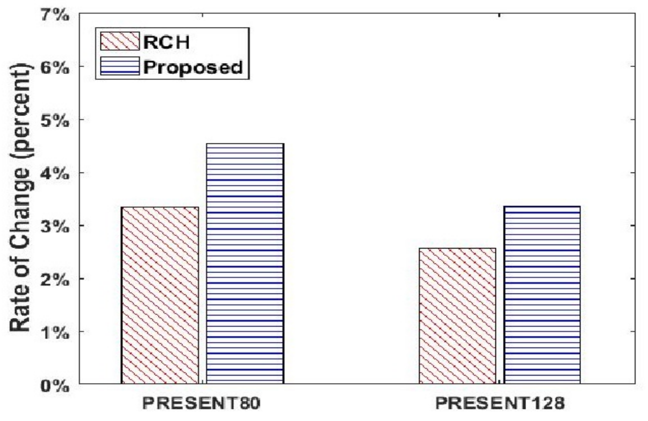
Fig. 1. Comparison of the execution time of original ciphers, the RCH simulations and the round order and internal parameters of algorithms randomization based ciphers in terms of the 80-bit and 128-bit keys for (a) PICCOLO, (b) TWINE and (c) PRESENT.



(a)



(b)



(c)

Fig. 2. Comparison of the energy consumption of original ciphers with the RCH method and the round order and internal parameters of algorithms randomization based ciphers in terms of the 80-bit and 128-bit keys for (a) PICCOLO, (b) TWINE and (c) PRESENT.

In Figure 1, a comparison is made between the execution time of the original ciphers, the Reconfigurable Hardware (RCH) [18], and the proposed block ciphers. We consider 80-bit and 128-bit keys for PICCOLO, TWINE, and PRESENT algorithms for these different cases. As seen for PICCOLO with an 80-bit key, the differences in execution time between the original cipher and RCH for plaintext of 512, 1024, 2048, and 3072 are 3.11%, 3.69%, 3.89%, and 4.02% while those differences between our proposed ciphers and the original cipher are 8.87%,9.64%, 10.06%, and 10.51%.

Furthermore, for 128-bit of PICCOLO the increases in time for 512, 1024, 2048, 3072 bit plaintexts are 4.51% 4.63%, 4.51%, and 4.70%, respectively, whereas the increases in time for our method against the original cipher are 10.98%, 11.82%, 12.35%, and 12.48%. The results of the same comparison between the original ciphers and the RCH method for TWINE80 with plaintext 512, 1024, 2048 and 3072 are 26.35%, 26.39%, 26.38%, and 26.38% while the results for our methods are 27.05%, 27.10% ,27.10%, and 27.11% for the same plaintext. For TWINE128 the rates of change from original cipher in RCH mode for 512, 1024, 2048, and 3072 plaintexts are 24.81%, 24.82%, 24.83%, and 24.83%, in the meantime, those measurements for our method are 25.53%, 25.55%, 25.56%, and 25.57%. The similar results were obtained for PRESENT algorithms. The differences in execution time for PRESENT80 and PRESENT128 are shown in Figure 1(c). It should be noted that the differences between the proposed block ciphers and the RCH method are similar to previous results between the original ciphers and the RCH results.

One of the several approaches to compute energy consumption is using the CPU's operating voltage and the average current dragged by each cycle, which can be as [20]:

$$E = I \times N \times \tau \times V \tag{1}$$

where, I, N, τ, and V are the average current, the number of clock cycles, the clock period, and the voltage, respectively. Figure 2 compares the energy consumption of PICCOLO, TWINE, and PRESENT block ciphers. The supply voltage and average current of Cortex-M4 microcontrollers are 3.6 V and 0.0155 A, respectively. Its operating frequency is 84 MHz. Since the voltage, average current, and clock period are constant, the number of clock cycles for each encryption round should be calculated and compared, which are provided by Data Watchpoint and Trace (DWT) [15]. As can be seen, the differences of average clock cycles of PICCOLO80 and PICCOLO128 between the original cipher and the RCH method are 3.86% and 4.61%, respectively. On the other hand, the increases of PICCOLO's average clock cipher compared with our round order and internal values randomization-based ciphers are only 5.27% and 5.92%, respectively. In the same way, the increases in average clock cycles for RCH are 26.38% and 24.82% for TWINE80 and TWINE128, respectively. In PRESENT80 and PRESENT128, the change rates of average clock cycles for reconfigurable hardware (RCH) are 3.33% and 2.57%. In the meantime, those increment rates for our method are 4.54% and 3.35%. The same increases are shown in our results when compared with previous results. The amount of processed data in a period of time can be measured by throughput, which determines that lightweight block cipher has the best performance in an IoT environment [15]. First, we divide the number of cycles by the block size of each algorithm. The total encryption cycles per bit for each algorithm is obtained as [15]:

$$Encryption(cycles/bit) = \frac{Numberofcycles}{Blocksize} \tag{2}$$

Since MCU operates under 84MHz, it can execute 84,000,000 cycles in each second. Therefore, the throughput of each block

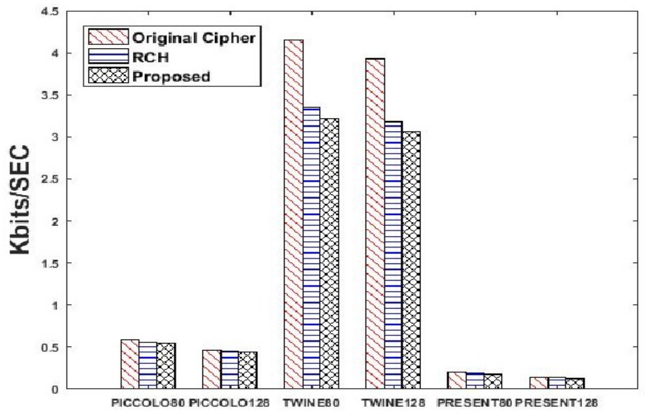


Fig. 3. Throughput comparison of original ciphers, the RCH simulations and the round order and internal parameters of algorithms randomization based ciphers in terms of the 80-bit and 128-bit keys for PICCOLO, TWINE, and PRESENT.

cipher can be expressed as follows [15]:

$$Throughput = \frac{CPUspeed}{Enc(cycles/bit)} \tag{3}$$

In Figure 3, a comparison is made between the throughput of original ciphers, the RCH simulations, and the round order and internal parameters of algorithms randomization-based ciphers in terms of the 80-bit and 128-bit keys for PICCOLO, TWINE, and PRESENT. As can be seen, all throughput values are close together for each cipher.

IV. Cryptanalysis of the Proposed Reconfigurable LWBCs

Most security analysis has been considered for proposed randomization algorithms against critical attacks such as the differential attack, boomerang attack, and Meet In The Middle attack (MITM). The differential attack is a chosen-plaintext attack that analyzes how the difference of input evolves through the several rounds of the cipher. To be exact, the probability of observing the difference of output (δ_{out}) that gives an input difference (δ_{in}) is as follows [21]:

$$\Pr[f(x \oplus \delta_{in}) \oplus f(x) = \delta_{out}] \tag{4}$$

In the following, a block cipher (E) reduced to t rounds will be denoted by E^t, as shown in Figure 4(a). As a result, $Pr[\delta_0 \rightarrow \delta_t]$ represents the probability of a differential $\delta_0 \rightarrow \delta_t$:

$$\Pr[\delta_0 \rightarrow \delta_t] = \Pr_{X,K}[E^t_K(X) \oplus E^t_K(X \oplus \delta_0)] \tag{5}$$

where $Pr_{X,K}$ is the probability computed on all possible input plaintext (X) and all possible keys (K). This probability can be calculated as:

$$\begin{aligned} Pr_{X,K}[E^t_K(X) \oplus E^t_K(X \oplus \delta_0)] &= Pr_{X,K}[C \oplus (C \oplus \delta_t)] \\ &= Pr_{X,K}[\delta_t] \end{aligned} \tag{6}$$

For the proposed reconfigurable block ciphers, the second time it will be a different block cipher with a different key after the

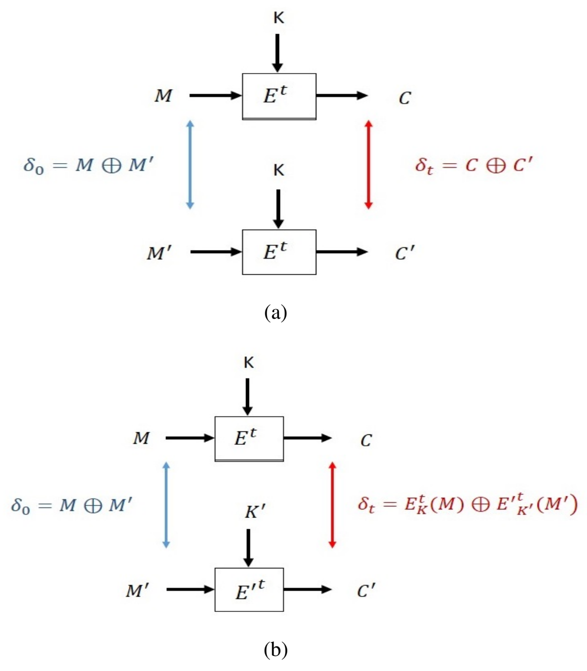


Fig. 4. A differential on t rounds of (a) cipher E and (b) our round order and internal parameters randomization based ciphers.

block cipher is run for r rounds. In this way, the probability of all possible input plaintexts X and key K is as follows:

$$Pr_{X,K}[E_K^t(X) \oplus E'^{\,t}_{K'}(X \oplus \delta_0)] = \Pr_{X,K}[C \oplus (C' \oplus \delta_t)] \neq \Pr_{X,K}[\delta_t] \quad (7)$$

The difference between outputs and inputs for block ciphers is depicted in Figure 4(b). The boomerang attack is a chosen-plaintext and cipher text attack which maximizes the probability of breakage by combining sets of four messages of M_0, M_1, M_2 and M_3. This attack uses differentials like a differential attack for examined algorithms [22] [23]. We implement this method for one of the messages, the result is the same for the other ones. Since the equation of probability is not equal to the difference of messages (α), our method has high safety against this attack as follows:

$$\begin{aligned} &E^{-1}(E(M_0) \oplus \delta) \oplus E'^{-1}(E'(M_0 \oplus \alpha) \oplus \delta) \\ &\Rightarrow [M_0 \oplus E^{-1}(\delta)] \oplus [M_0 \oplus \alpha \oplus E'^{-1}(\delta)] \\ &\Rightarrow E^{-1}(\delta) \oplus E'^{-1}(\delta) \oplus \alpha \neq \alpha \end{aligned} \quad (8)$$

The newest method of attack for this category is the three-subset Meet In The Middle (MITM) attack, which has good results on lightweight block ciphers [24] [25]. In PICCOLO algorithms, the number of rounds is considered variable. Therefore PICCOLO80 and PICCOLO128 have 10 and 16 possible number of rounds. The MITM attacks have two sides. The right side starts encryption operation partially from the beginning and the left side performs decryption partly from the ending. For PICCOLO algorithms, the right and left sides equations are performed for each guess of the subkeys, respectively, as follows:

$$v = \lambda_{1,i}(p) = k_{1,i} \oplus CON_{1,i} \quad (9)$$

$$u = \lambda^{-1}_{i+1,r}(c) = k_{i+1,r} \oplus CON_{i+1,r} \quad (10)$$

where $\lambda_{i,j}$ describes the operation of an r-round block cipher encryption (from round i to round j) with a fixed key and $\lambda_{i,j-1}$ describes the decryption in the same circumstances. The key schedule and constant value of rounds are indicated by $k_{i,j}$ and $CON_{i,j}$. The computational complexity which is indicated by $\varsigma_{comp\ origind}$ will be reduced with respect to the MITM attack. In our algorithms, due to the randomization of key scheduling part and constant value of rounds, the computational complexity is increased. For 80-bit and 128-bit keys, the complexity is computed as follows:

$$\varsigma_{compP80} = 10 \times 5! \times 30! \times \varsigma_{comporiginal} \quad (11)$$

$$\varsigma_{compP128} = 16 \times 8! \times 41! \times \varsigma_{comporiginal} \quad (12)$$

For TWINE algorithms, the right and left sides of the MITM attack are performed for each guess of subkeys as follows:

$$v = \lambda_{1,i}(p) = k_{1,i} \oplus S_{1,j} \oplus CON_{1,i} \oplus \pi_{1,j} \quad (13)$$

$$u = \lambda^{-1}_{i+1,r}(c) = k_{i+1,r} \oplus S_{i+1,r} \oplus CON_{i+1,r} \oplus \pi^{-1}_{i+1,r} \quad (14)$$

where $k_{i,j}$, $S_{i,j}$, $CON_{i,j}$, $\pi_{i,j}$ and $\pi^{-1}_{i,j}$ are key schedule, S-box, constant value of rounds, and diffusion of block indexes for encryption and decryption, respectively. In proposed algorithms for TWINE, because of randomizing the key scheduling part, S-box, constant value of rounds and diffusion of block indexes for encryption or decryption, the computational complexity is increased. For the proposed TWINE with 80 and 128 bits keys this can be calculated as:

$$\varsigma_{compT80} = 5! \times 16! \times 36! \times 16! \times \varsigma_{comporiginal} \quad (15)$$

$$\varsigma_{compT128} = 8! \times 16! \times 36! \times 16! \times \varsigma_{comporiginal} \quad (16)$$

The mentioned MITM equations for PRESENT algorithms are as follows:

$$v = \lambda_{1,i}(p) = k_{1,i} \oplus Sbox_{1,j} \oplus Player_{1,i} \quad (17)$$

$$u = \lambda^{-1}_{i+1,r}(c) = k_{i+1,r} \oplus Sbox_{j+1,r} \oplus Player_{i+1,r} \quad (18)$$

where $k_{i,j}$, $S_{i,j}$, and $P - layer_{i,j}$ are key scheduling part, S-box layer, and the P-layer. In the proposed randomization algorithms for PRESENT, the increase in complexity for both key sizes is the same and it is computed as follows:

$$\varsigma_{compPRESENT} = 12 \times 64! \times 32! \times 16! \times 64! \times \varsigma_{comporiginal} \quad (19)$$

V. Conclusion

In this paper, the security performance of lightweight block ciphers including the PICCOLO, TWINE, and PRESENT, using randomization round order and internal parameters of algorithms are presented for IoT environments. Our results indicate that the proposed reconfigurable-based block ciphers exhibit significant improvements in security performance with minor and negligible changes in energy-throughput performances. In other words, the round order and internal parameters randomizations have minimal effect on the complexity of the lightweight block ciphers, but they significantly decrease an attacker's ability to guess keys.

References

[1] M. Maroufi, R. Abdolee, and B. M. Tazekand, "On the convergence of blockchain and internet of things (IoT) technologies," *Journal of Strategic Innovation and Sustainability (JSIS), vol. 14*, pp. 1–11, 2019.

[2] M. N. Bhuiyan, M. M. Rahman, M. M. Billah, and D. Saha, "Internet of things (IoT): A review of its enabling technologies in healthcare applications, standards protocols, security and market opportunities," *IEEE Internet of Things Journal*, 2021.

[3] J. Yogi, U. S. Chauhan, A. Raj, M. Gupta, and S. S. Sudan, "Modeling simulation and performance analysis of lightweight cryptography for IoT-security," in *2018 3rd International Conference and Workshops on Recent Advances and Innovations in Engineering (ICRAIE)*. IEEE, 2018, pp. 1–5.

[4] K. McKay, L. Bassham, M. Sönmez Turan, and N. Mouha, "Report on lightweight cryptography," National Institute of Standards and Technology, Tech. Rep., 2016.

[5] U. du Luxembour. Lightweight block ciphers. [Online]. Available: https://www.cryptolux.org/index.php/Lightweight-Block-Ciphers

[6] X. Fan, K. Mandal, and G. Gong, "WG-8: A lightweight stream cipher for resource-constrained smart devices," in *International Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness*. Springer, 2013, pp. 617–632.

[7] K. Iokibe, K. Maeshima, H. Kagotani, Y. Nogami, Y. Toyota, and T. Watanabe, "Analysis on equivalent current source of AES-128 circuit for HD power model verification," in *2014 International Symposium on Electromagnetic Compatibility, Tokyo*. IEEE, 2014, pp. 302–305.

[8] R. L. Rivest, "The RC5 encryption algorithm," in *International Workshop on Fast Software Encryption*. Springer, 1994, pp. 86–96.

[9] J. Yu, G. Khan, and F. Yuan, "XTEA encryption based novel RFID security protocol," in *2011 24th Canadian Conference on Electrical and Computer Engineering (CCECE)*. IEEE, 2011, pp. 000 058–000 062.

[10] G. Leander, C. Paar, A. Poschmann, and K. Schramm, "New lightweight DES variants," in *International Workshop on Fast Software Encryption*. Springer, 2007, pp. 196–210.

[11] 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 *International workshop on cryptographic hardware and embedded systems*. Springer, 2007, pp. 450–466.

[12] J. Hosseinzadeh and M. Hosseinzadeh, "A comprehensive survey on evaluation of lightweight symmetric ciphers: hardware and software implementation," *Advances in Computer Science: an International Journal*, vol. 5, no. 4, pp. 31–41, 2016.

[13] B. J. Mohd, T. Hayajneh, and A. V. Vasilakos, "A survey on lightweight block ciphers for low-resource devices: Comparative study and open issues," *Journal of Network and Computer Applications*, vol. 58, pp. 73–93, 2015.

[14] D. J. Wheeler and R. M. Needham, "TEA, a tiny encryption algorithm," in *International workshop on fast software encryption*. Springer, 1994, pp. 363–366.

[15] L. Ertaul and S. K. Rajegowda, "Performance analysis of CLEFIA, PICCOLO, TWINE lightweight block ciphers in IoT environment," in *Proceedings of the International Conference on Security and Management (SAM)*. The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 2017, pp. 25–31.

[16] K. Shibutani, T. Isobe, H. Hiwatari, A. Mitsuda, T. Akishita, and T. Shirai, "PICCOLO: An ultra-lightweight blockcipher," in *International workshop on cryptographic hardware and embedded systems*. Springer, 2011, pp. 342–357.

[17] R. Abdolee and V. Vakilian, "Reconfigurable security hardware and methods for internet of things (IoT) systems," Oct. 29 2020, US Patent App. 16/859,478.

[18] S. A. Kahan, R. Abdolee, E. Argueta, and V. Vakilian, "Security performance improvement of lightweight block ciphers via round order randomization," in *International Conference on Communication and Signal Proccessing (ICCSP), Tehran*. IEEE, 2018, pp. 1–99.

[19] Systemy RT I embedded. [Online]. Available: http://www.ue.pwr.wroc.pl/systemy-rt/RTE6.pdf

[20] D. Salama, H. A. Kader, and M. Hadhoud, "Studying the effects of most common encryption algorithms," *International Arab Journal of e-Technology*, vol. 2, no. 1, pp. 1–10, 2011.

[21] E. Biham and A. Shamir, *Differential cryptanalysis of the data encryption standard*. Springer Science & Business Media, 2012.

[22] D. Wagner, "The Boomerang attack," in *International Workshop on Fast Software Encryption*. Springer, 1999, pp. 156–170.

[23] J. Kelsey, T. Kohno, and B. Schneier, "Amplified Boomerang attacks against reduced-round MARS and Serpent," in *International Workshop on Fast Software Encryption*. Springer, 2000, pp. 75–93.

[24] K. Aoki and Y. Sasaki, "Meet-In-The-Middle preimage attacks against reduced SHA-0 and SHA-1," in *Annual International Cryptology Conference*. Springer, 2009, pp. 70–89.

[25] A. Bogdanov and C. Rechberger, "A 3-subset Meet-In-The-Middle attack: cryptanalysis of the lightweight block cipher KTANTAN," in *International Workshop on Selected Areas in Cryptography*. Springer, 2010, pp. 229–240.