Lightweight Countermeasure to Differential-Plaintext Attacks on Permutation Ciphers

Matthew Lewandowski, Srinivas Katkoori

Lightweight Countermeasure to Differential-Plaintext Attacks on Permutation Ciphers

Matthew Lewandowski and Srinivas Katkoori

University of South Florida, Tampa FL 33620, USA,
{mlewando,katkoori}@mail.usf.edu

Abstract. Many lightweight permutation based block ciphers have emerged for the use of encryption and security in the Internet of Things (IoT). However, recent work has shown a critical vulnerability in these ciphers due to the employment of static permutation networks in their architectures; Side-Channel Analysis and Differntial-Plaintext Attack (SCADPA) can be effectively performed on any cipher utilizing permutation networks. In this work, we present a lightweight solution for combating SCADPA. We demonstrate how this countermeasure can be employed on existing ciphers using the lightweight IoT PRESENT-80 cipher, providing multiple architectural implementations, and comparing the performance of these modified architectures against the unaltered PRESENT-80 cipher. Additionally, we analyze how this countermeasure impacts the resilience for all permutation ciphers when considering this attack scheme and provide alternative implementations and possible enhancements.

Keywords: Ciphers, Countermeasures, Differential-Plaintext Attack, Encryption, Permutation Networks, Side-Channel Analysis, SCADPA

1 Introduction

As Integrated Circuit (IC) technologies continuously advance we have seen many computing devices (μcomputer, μcontroller, etc...) being made affordably and widely available to the public; driving curiosity & innovation for many through rapid prototyping, deployment, or even exploitation, of hardware systems. Similarly, the technology and capabilities of consumer goods has also largely grown over the decades; for instance, toys, cars, garage doors, washers & dryers, are all some of the commonplace devices that are now being equipped with embedded systems and wireless communication capabilities for enhancing the end-user experience. However, as technology continued to advance and grow so too did the vastness of information that was becoming readily available and accessible by nearly everyone around the world, and at the touch of their fingertips. As a result, it has become possible for many without prior knowledge or experience to adequately obtain the skills necessary for designing and implementing hardware systems that can enhance or exploit everyday technology.

Even though the devices that contribute to this Internet of Things (IoT) were intended for improving quality-of-life and furthering the smart home revolution, they have since become a common targets among many hardware hackers; with home devices being targeted for vulnerability, enhancement, or simply repurposing. While it may be understandable that many of these home devices (toys, washer, etc...) don't necessarily require proper access & control mechanisms, it would be expected that devices otherwise designed for the purpose of access control (garage door) would implement some form of secure communication protocol. However, websites such as YouTube have become commonplace for hardware hackers to demonstrate exploitation of these IoT devices that we otherwise naïvely expect to keep our homes secure from others; videos for cloning Radio Frequency Identification (RFID) security badges, hacking: electronic home locks, magnetic door locks, wireless doorbells, garage doors, and even cars, can all be found. While this can be seen simply as an innocent dissemination of knowledge and educational content, its darker side exposes the general lack of concern for security by device manufacturers and existing susceptibilities of systems with respect to the ever evolving technology; allowing for those with ill-intent to easily and affordably construct malicious systems that can be used for personal gain.

In this work we present a practical and lightweight design method for incorporating key-dependent substitution and permutation networks within cryptographic algorithms and hardware through Interconnection Network (ICN) primitives. This design method is demonstrated via modification to the PRESENT-80 cipher with several architectural variations to demonstrate flexibility. Additionally, we show that, for this cipher, incorporation of this key-dependent design methodology has little-to-no impact on resulting performance or power results obtained during synthesis and only serve to increase the resilience while acting as a countermeasure to the critical susceptibility exhibited by permutation based ciphers.

While ICNs have been shown to be applicable to cryptosystems, the novelty of this work stems from the previously unexplored applications of ICNs at the scope of a single switch rather than just providing methods of permutation via routings (omega, baseline, etc...); exploring how resilience of cryptosystems can be enhanced via switching networks and do not require the use of additional complex mathematical operations in-order to provide security. Further, we provide viable countermeasures and discuss how this design methodology can be applied to alternative key methods or even constructing ciphers solely from this methodology.

The rest of this work is organized as follows: Section 2 provides necessary background on cryptographic fundamentals, ICNs, and the attack model considered. Section 2.5 details related works relevant to the approach presented in Section 3, where Section 4 provides experimental results compared against an unaltered implementation of the PRESENT-80 cipher. Section 5 performs an in-depth analysis of the presented approach with respect to existing attacks on

the cipher and implications of this design methodology against these attacks. Finally in Section 6 we draw conclusions.

2 Background and Related Work

In this section we cover the necessary background information and related works relevant to the proposed approach; beginning by covering the basics of information security and follow with the attack model considered throughout this work. We then provide information relevant to the proposed approach and review existing works that present countermeasures to differential-plaintext attacks.

2.1 Cryptography & Information Security

The National Institute of Standards and Technology (NIST) defines what may now be commonly referred to as the *CIA Triad* [14], and is composed of the three fundamental services required for information security: (1) Confidentiality, (2) Integrity, (3) Availability [7]; simply requiring: (1) privacy and protection of data from unauthorized parties, (2) the data may not be altered by unauthorized parties, and (3) the the confidential data to be readily accessible and in a timely manner. However, it is commonly believed that information security is incompletely represented by this triad alone, requiring additional services: (4) Authenticity and (5) Non-repudiation [14]; simply requiring: (4) that data be verifiable as genuine, and (5) that data/actions can be adequately linked to the appropriate originating entity.

While this set of services are basic principles & requirements for information security they are also applicable to cryptography and must be upheld. However, behind any and all cryptographic algorithms are Shannon's [13] properties of (1) confusion and (2) diffusion; more simply, (1) the resulting ciphertext should be dependent on more than one part of the key and (2) changing a single bit in the plaintext should produce a larger change in the resulting ciphertext. This small alteration to the plaintaxt that results in a larger change in the ciphertext is known as the *avalance effect*. *Confusion*, commonly performed using the Substitution-Box (S-Box), replaces blocks of bits in the plaintext with some other block, e.g., the character c is replaced by a. *Diffusion*, potentially using the Permutation-Box (P-Box), permutes the plaintext,e.g., *cat* becomes *tca*. These two concepts ultimately relate to the integration of substitution and transposition for the plaintext to ciphertext transformation through some involved process.

2.2 PRESENT-80 Cipher

In order to provide a better relational understanding of the attack model to permutation based ciphers, here we will first detail the PRESENT-80 cipher, as first presented by [3]. The 80-bit implementation of the cipher utilizes an 80-bit secret key, wherein only 64-bits of the key are used each round and in conjunction

with an additive value from the round counter; as 32 rounds are performed on the plaintext data. This key-schedule is illustrated by Figure 1 and depicts the round subkey (64 of 80-bits) and the update operation for constructing the next round subkey; additionally, the 5-bit Exclusive OR (XOR) operation depicts the additive round counter data. The round architecture is shown by Figure 2, wherein the series of boxes labeled with S denote the SBox operations, detailed by Table 1, with the subsequent routing depicting the PBox operation.

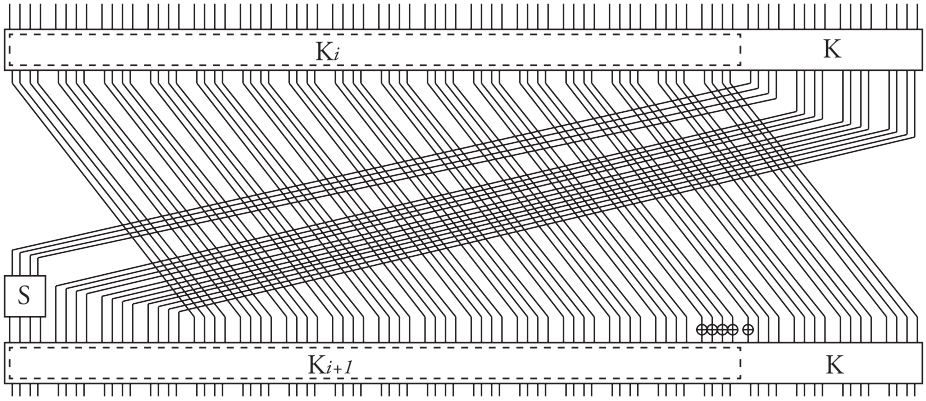


Fig. 1. PRESENT-80 Key Schedule

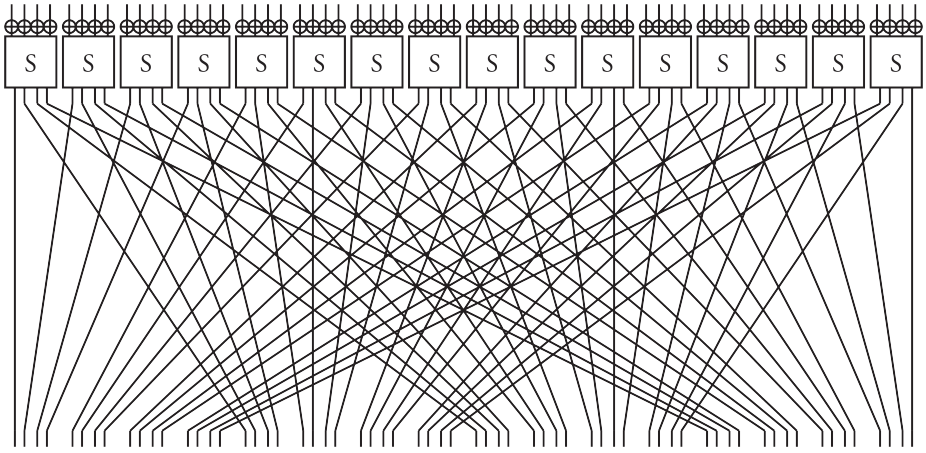


Fig. 2. PRESENT-80 Round Architecture

Table 1. PRESENT-80 Substitution 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

2.3 Attack Model

The attack model we consider here is that first presented in [4], which constructs an attack method on the permutation network of the PRESENT-80 cipher which is ultimately shown applicable to all permutation based ciphers.

The SCADPA methodology utilizes a differential-plaintext attack and is initially performed by capturing power/Electromagnetic (EM) leakage in order to obtain SBox differential at the round output. To better illustrate this consider the following example, exemplified by Figure 3, wherein if we consider any single bit change to the leftmost nibble then the resulting bits/locations that are potentially always impacted by this change are shown in red.

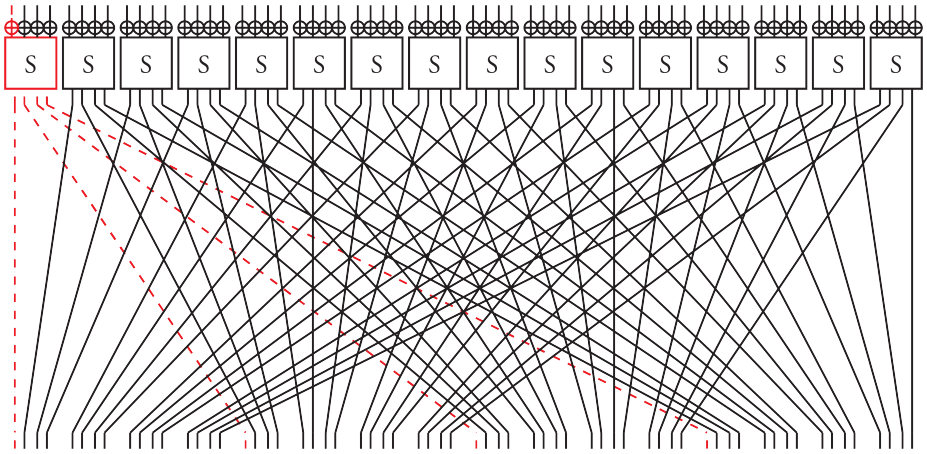


Fig. 3. Bits affected from changing a single bit input

The SCADPA method serves to exploit this architectural design limitation/convention, and as to the best of our knowledge, all SPN based ciphers utilize such an exploitable design methodology; thus, the bits that are potentially affected by changing a single nibble will always be in the same position.

2.4 Interconnection Networks

ICNs can be used for a variety of routing applications: network switching, high-performance/parallel computing, and even sorting. Non-blocking IDNs are networks wherein the endpoints are outside (indirect) and that any permutation of source-destination is possible via switching (non-blocking). The most basic non-blocking indirect switching element, shown by Figure 4, is known as the *Banyan Switch* and is the foundation for intermediate flow-control of data for a variety of network topologies. For clarity, we note that *Contention Free* in Figure 4 denotes that no two starting endpoints will attempt to utilize the same route. Additionally, the focus of the focus of this work will not be on the routing topologies between these Banyan structures, rather, it will focus on the use of solely these Banyan structures within cryptographic architectures.

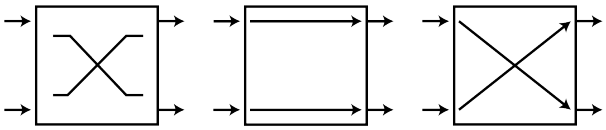


Fig. 4. Contention Free Banyan Switch

2.5 Interconnection Networks in Crpyotgraphy

Several works [5, 15, 9] have been proposed which explore the use of ICNs outside telecommunication systems for producing low latency permutations. Similarly, only a limited number of works [11, 8, 16] appear to have been proposed exploring ICNs in cryptographic algorithms.

The first work [11] that appears to present the concept of integrating ICNs into cryptographic architectures does so by demonstrating the design of a one-key cryptosystem using a combination of ICNs and boolean functions. For clarity, a one-key system is that of a turn dial padlock, i.e., only one key will open the lock. To combat contention issues in switching elements this work utilizes a *control-setting function* (h) for controlling passthrough and crossover operations of Banyans. This function h, generated from some set of boolean operations, controls all Banyans in the topology; further showing that h can be enhanced through the use of a pseudo-random function generator. Ultimately this work shows that the use of ICNs in cryptography is a viable method for performing permutations while fulfilling the *avalanche* property.

Another work [8] shows that integration of ICNs for performing permutations in cryptographic software can successfully reduce the number of instructions and cycles needed in n-bit permutations. This is achieved through the development of four different permutation methods, wherein they show that each of these methods reduce cycle & instructions when compared to using Lookup-table (LUT) primitives; thus further showing that the application of network primitives can be advantageous in cryptosystems. A significant difference between this work ([8]) and [11] is how controlling bits for switching elements are generated, unlike that of [11] a variety of methods generate control sequences and switching element functionality is extended to handle *don't care* control values. Additionally, this method doesn't implement at rue switching network per say, it utilizes a series of registers with routing topologies (benes, butterfly, etc...) and connects register locations with 2-to-1 multiplexers; the output selected by the control value is passed to a temporary register.

More recently, [16] presents an On-Chip-Network (OCN) specifically targeting hardware based cryptographic cores. This work utilizes what is known as a *wormhole* ICN which is based on a *Flit* control method, rather than function generators. Flits are essentially data contained within a packet of information; with the format of a packet as: $\{Head, Flit, Flit, ..., Tail\}$. The wormhole ICN is simply a large mesh based ICN where communication of packets is done through single pieces of a packet and in a manner where the tail follows flits which follow

the head that is initially sent through the wormhole. Conversely, as previous methods [11, 8] utilize non-blocking networks, a wormhole is inherently blocking and does not have contention free routing. In the event that two packets attempt to utilize the same route then one will be blocked and all subsequent pieces following the head will also stop. Additionally, this method explores the use of both randomized and pipelined scheduling for burst encryption of data. Results provided from the randomized scheduling method further enforce the fact that collision/contention is not handled in this system as [16] states that this method incurs longer transmission times due to increased collisions and disordering of packets; with the pipelined method presented to alleviate this drawback.

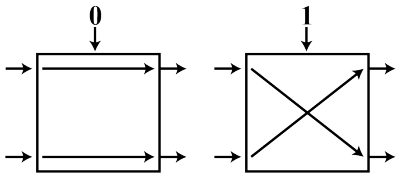
2.6 Key-Dependency In Cryptography

Several works exist in literature presenting the concept of key-dependent structures in cryptographic algorithms, [2, 10]. However, they currently only pertain to, and demonstrate, enhanced security and avalanche properties for dynamic key-dependent Substitution Box (SBox) networks. While enhancing SBox resilience via key-dependent mechanisms provides enhanced security in one aspect it still does not aim to address other susceptibilities of many algorithms based on the previously detailed attack model.

3 Key-Dependent Interconnection Network Structures

As mentioned, existing key-dependent implementations do not aim to address the most recent susceptibilities in cryptographic algorithms, for this reason we aimed to provide a lightweight mechanism that can produce both key-dependent SBox and PBox networks. Thus, in this section we introduce the proposed approach for combating such differntial-plaintext attacks applicable to not only IoT based, but all permutation based ciphers.

The proposed approach utilizes a controlled version of the Banyan switch for inducing key-dependent permutations of round data in PRESENT-80; however, this approach is still applicable to any cipher using permutation networks, not just PRESENT-80. To better illustrate this we depict how the operations of a controlled Banyan switch via Figure 5.



(a) 0 = Pass (b) 1 = Crossover

Fig. 5. Controlled Banyan Switch

Following, the architecture shown by Figure 6(a) was constructed based on the original presentation and detailing of the PRESENT-80 cipher by [3], and subsequent architectures Figure 6(b)-6(d) are those modified to incorporate a layer of controlled Banyan structures (*bLayer*), implemented as shown by Figure 6(e), at various points in the round architecture. For simplicity, the bLayer was constructed around the PRESENT-80 cipher, such that, the input/output bus widths of the Banyan switches was based on the number of Banyans that were easily integrated: 16, as 16-bits of the 80-bit key went unused per round.

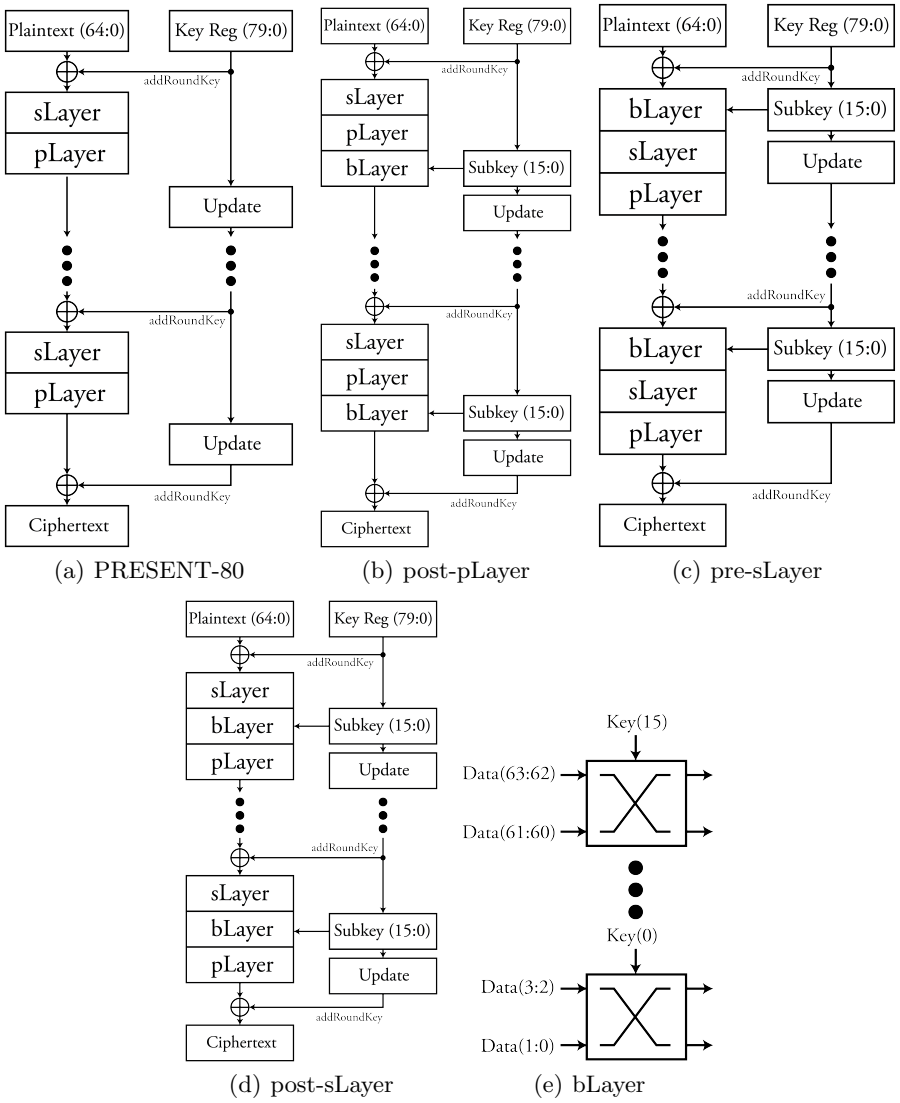


Fig. 6. PRESENT-80 Architectures Examined & bLayer Implementation

4 Experimental Evaluation

The architectures, shown by Figure 6, were constructed and verified using Xilinx ISE 14.6, VHDL-93, and the Nexys 3 development board from Digilent (XC6SLX16-3CSG324). The remainder of this section, we provide results in terms of performance, area, and power data gathered during the synthesis process for each of the architectural implementations, while comparing it against the unaltered PRESENT-80 cipher.

We first present the synthesis results for each of these architectures in Table 2, where Lookup-Tables (LUTs) and Registers are reported in terms of Slice Logic Utilization as reported by Xilinx ISE when utilizing the default synthesis design goal & strategy: Balanced, Xilinx Default (unlocked). Additionally, for clarity, the parenthetically enclosed percentages reported in Table 2 denote the overall percentage of slice logic usage for the Nexys 3 development board used; the total number of slices for both LUTs and registers for this package are 9112 and 18224, respectively.

Table 2. Summary of Synthesis Results

Architecture	**MHz**	**LUTs**	**Registers**	%Δ M	%Δ L	%Δ R
PRESENT-80 (Figure 6(a))	223.821	589 (6%)	741 (4%)	–	–	–
Figure 6(b)	223.821	497 (5%)	757 (4%)	0	-15.62	2.16
Figure 6(c)	223.821	589 (6%)	741 (4%)	0	0	0
Figure 6(d)	223.821	482 (5%)	745 (4%)	0	-18.17	0.54

Following this, the architectures were mapped to the Nexys 3 board and power reports generated. Each of the designs was run with default signal activity for simplicity. Table 3 details the Total, Dynamic, and Static power for each of the PRESENT-80 architectures as reported by the Xilinx Power Estimator tool.

Table 3. Summary of I/O Power Results

Architecture	**Total**	**Dynamic**	**Static**	%Δ T	%Δ D	%Δ S
PRESENT-80 (Figure 6(a))	91.5 mW	69.3 mW	22.2 mW	–	–	–
Figure 6(b)	69.6 mW	47.58 mW	22.0 mW	-23.93	-31.34	-0.9
Figure 6(c)	91.5 mW	69.3 mW	22.2 mW	0	0	0
Figure 6(d)	75.0 mW	52.95 mW	22.1 mW	-18.03	-23.59	-0.45

For clarity, values less than zero reported in the %Δ columns of Table 2 and Table 3 denote improvements/reductions while those greater than zero indicate deterioration/increase of the corresponding metric.

4.1 Analysis of Results

Here we provide possible explanations for the results reported for both synthesis and power; as it can be seen that in nearly all cases modification of the original architecture to include the bLayer has shown to reduce power and slice usage. As this is otherwise counter-intuitive when considering hardware systems: more is generally not less. Thus, in the remainder of this section we will explore potential reasons for which the addition of logic produced both lower overall slice logic and power consumption.

When examining the Spartan-6 Configurable Logic Block (CLB) documentation, provided via Xilinx [1], more insight is given; specifically, the XC6SLX16 architecture has 9,112 LUTs. More importantly, these LUTs are specifically 6-input LUTs which is completely less obvious from the Xilinx ISE post synthesis report, as they are only reported as Slice LUTs. Now, consider Table 2, while a maximum difference of 1% in overall slice LUT usage appears minimal, or even negligible, when considering the $\%\Delta$ difference we can see that the overall change is relatively large in comparison. Thus, we hypothesize that additional bLayer logic allowed for the synthesis process to greater utilize the 6-input LUTs via logic reduction and optimization efforts provided by the strategy. This becomes more clear given Figure 7, the simplified implementation of an LUT6 [1], and consider how given outputs of a bLayer can be condensed into the logic of an LUT6 better than solely that of the logic for a static permutation network.

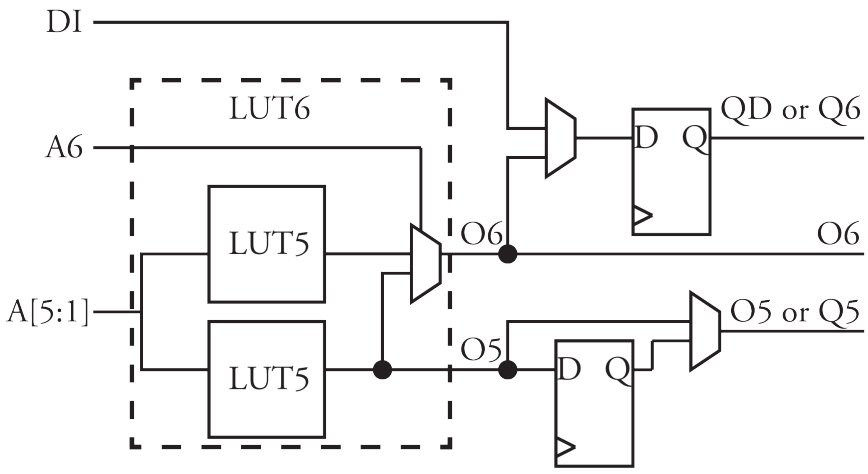


Fig. 7. Xilinx Spartan-6 Slice LUT Logic

Additionally, we secondarily hypothesize that through the reduction of 6-input LUTs and optimization of switching logic we also saw a reduction in total/dynamic/static power. We do note that further experiments utilizing other cryptographic algorithms and this method will need to be performed in order

to confirm/deny this hypothesis. In addition to this we believe that because slice logic differs between development boards and logic cores, that it must also be explored across multiple development boards and packages. Such that, this benefit is not exclusive to both the PRESENT-80 and Nexys 3/Spartan-6 FPGA/development boards.

5 Analysis

Here we examine concepts applicable to this work that aim to increase cipher resilience, how it can be applied to other ciphers, and existing works which further support the proposed approach.

5.1 Differential-Plaintext Attacks

Here we examine the implications of these additions and architectural modifications with respect to existing susceptibilities exhibited by the PRESENT-80 algorithm. We first examine one of the most recent works that not only attacks the PRESENT-80 cipher but is applicable to all bit permutation based ciphers, and is presented in [4].

This methodology, called SCADPA, utilizes a differential-plaintext attack and is initially performed by capturing power/Electromagnetic (EM) leakage in order to obtain SBox differential at the round output. However, while SCADPA captures leakage to capture entropy and determine SBox differential at the round output given that the permutation of bits is known, or static, consider this scenarios when examining the architecture presented by Figure 6(b).

If these permutations and observable round outputs were not static but rather a key-dependent outcome then the observable differential represents two possible values. For example, when using the bLayer, an output of '0001' for bits (63:60) of a given round has the potential to be either '0001' or '0100' based on key-dependency and controlled permutations. When considering the PRESENT-80 cipher this is significant, as the SBox is predetermined for 4-bit nibbles of round data, and is provided by Table 1. This means that based on such a key-dependent permutation the data substituted could have either originally been an 'E' or '3' and, thus, disrupts the ability to clearly ascertain an exact value of substitution from an otherwise static permutation network. Conversely, without use of the bLayer, a value of '0001' statically represented a substitution of '5' and allowed for a nibble of plaintext to be determined from the resulting permuted round data.

To best illustrate this, Figure 8 shows the potential bits that can be affected by changing the first nibble of plaintext in the cipher, conversely, by employing a key-dependent Banyan layer Figure 9 now illustrates the bits potentially affected, this increasing the number of possibilities and reducing the amount of static differential. For clarity, boxes labeled B in Figure 9 and subsequent figures denotes a controlled Banyan.

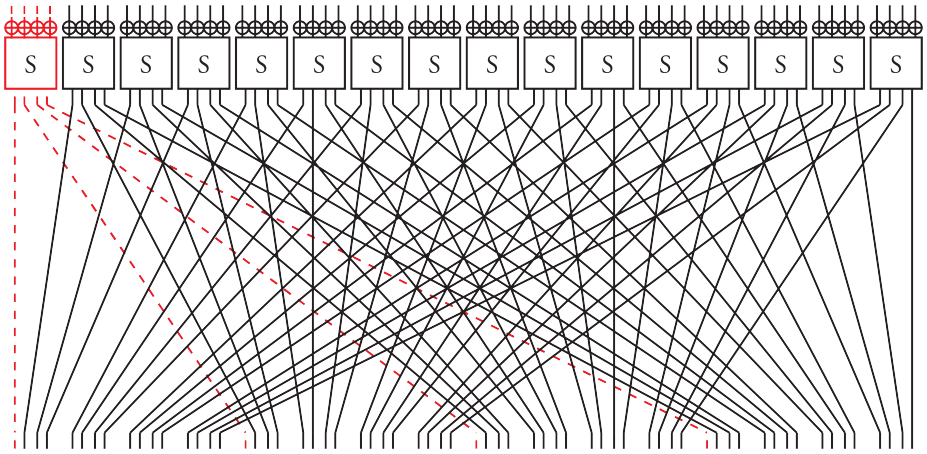


Fig. 8. Bits affected by changing first plaintext nibble

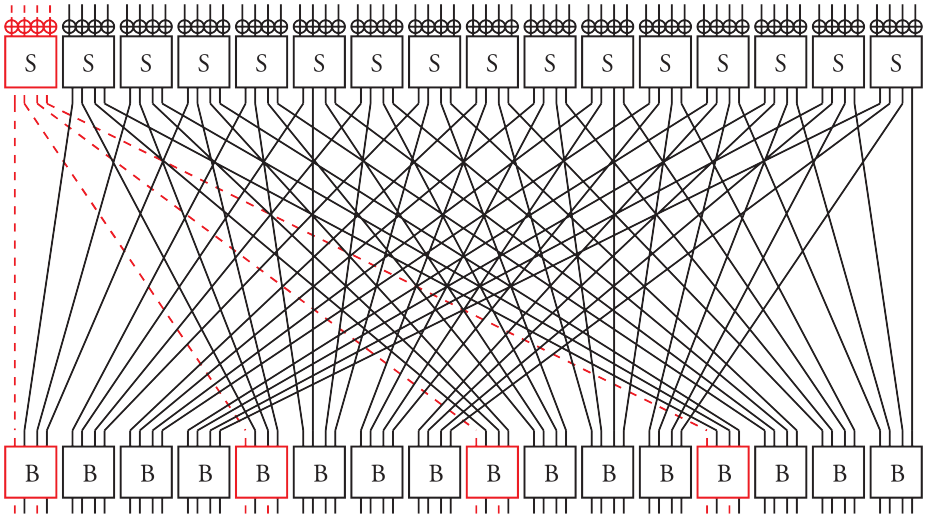


Fig. 9. Bits affected by changing first plaintext nibble when using bLayer for Figure 6(b); twice as many

Potential Improvements Due to the ordering of Banyan structures only roughly twice the number of bits are potentially affected; however, it is possible to implement an architecture which yields a larger number of bits being affected when changing the first nibble, which is illustrated by Figure 10. Similarly, through the use of bitwise Banyan operations the number of bits affected are five times as many, Figure 11, and round dependency can be integrated through utilization of Banyans and bits from the round counter Figure 12. For

clarity the five Banyan structures at the bottom in Figure 12 would be controlled by round counter bits.

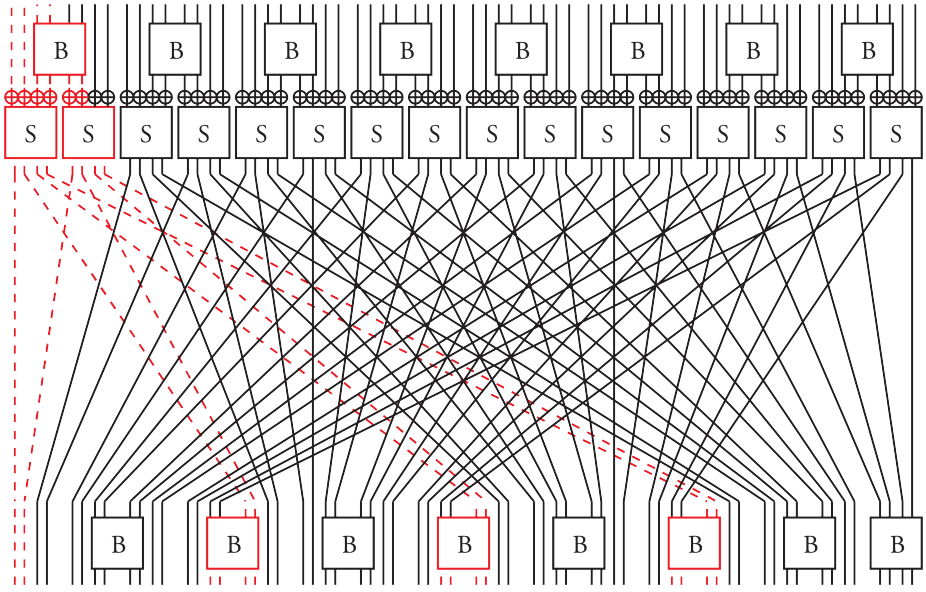


Fig. 10. Bits affected by changing first plaintext nibble; 3 times as many bits potentially affected

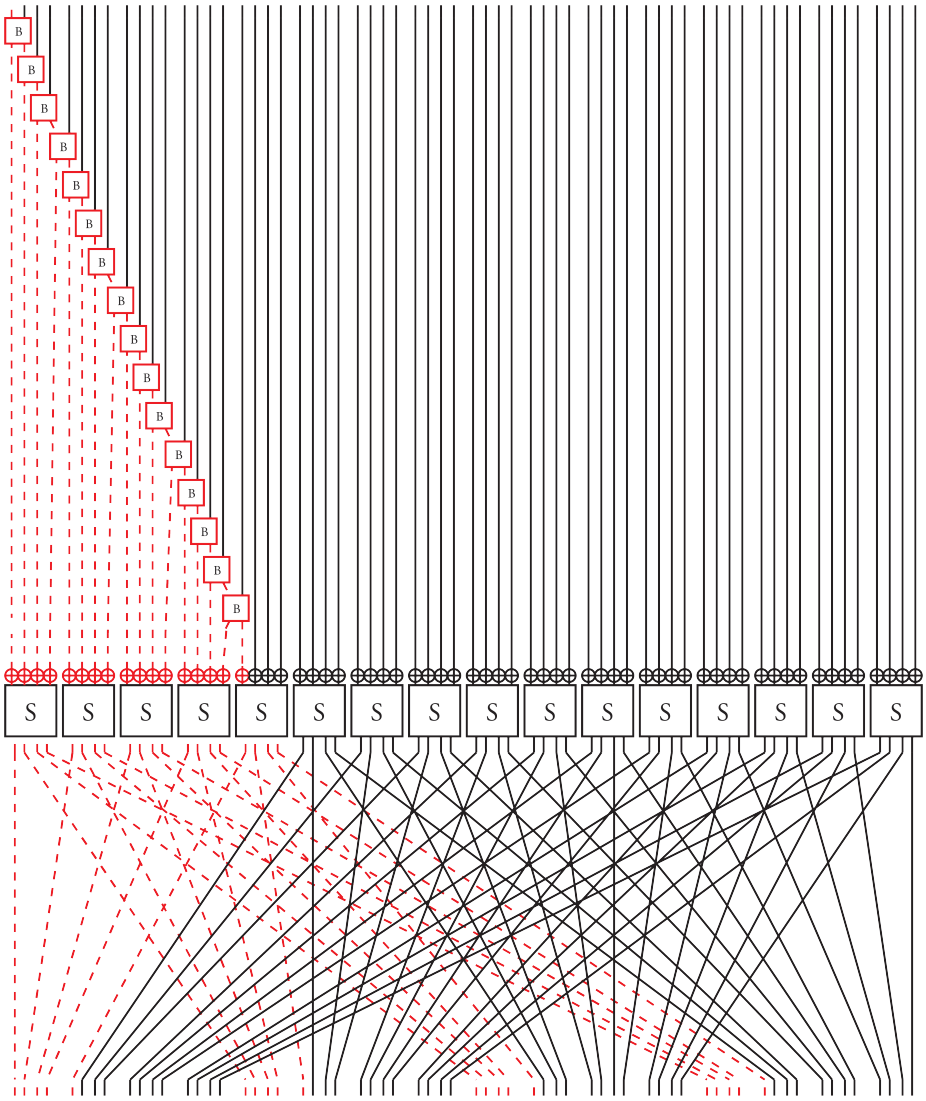


Fig. 11. "Waterfall" Banyan of Bits; 4 times as many bits potentially affected

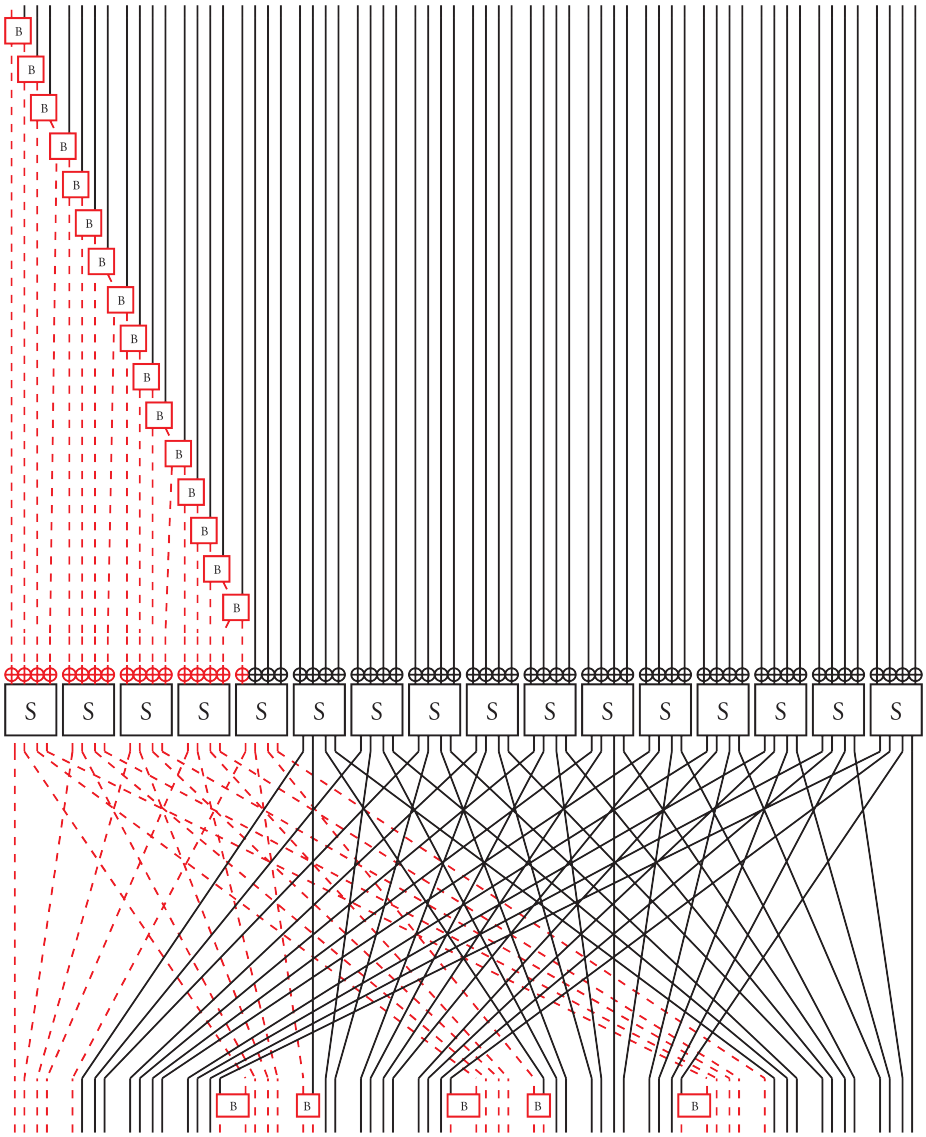


Fig. 12. "Waterfall" Banyan of Bits and Round Dependency; 6 times as many bits potentially affected

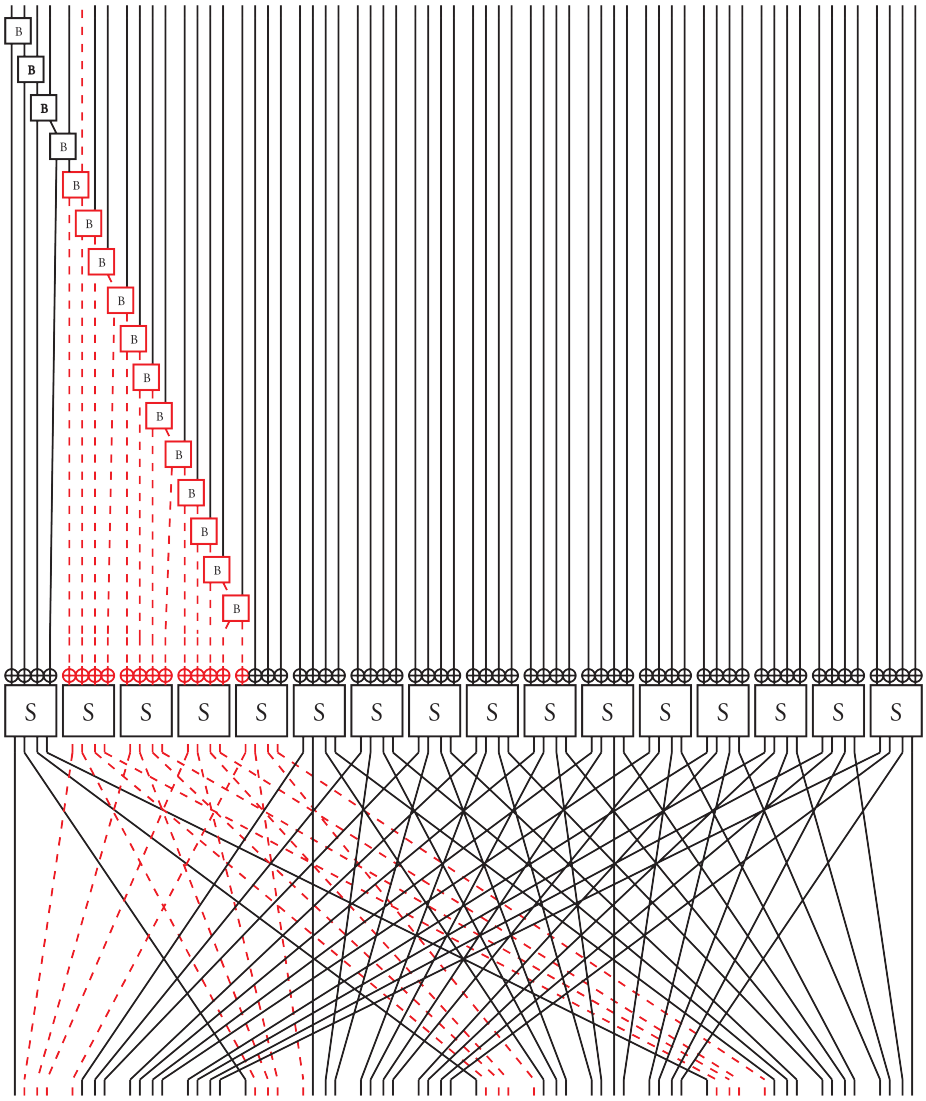


Fig. 13. "Waterfall" impact on Subsequent Bits, decreasing; 4 times as many bits potentially affected

5.2 Application to SBox

While the demonstration of this methodology was only applied to the PBox (pLayer) of the PRESENT-80 cipher it can easily be extended to enabling the use of key-dependent SBox permutations. If one were to apply controlled Banyan switches to some unique patterning of SBox data, where no collisions would occur for all combinations, then the incorporation of key-dependent SBox data is easily computed. For example, consider the two least significant bits in a

binary representation of hexadecimal 0-F, if a controlled Banyan is applied to these 16 numbers then a unique key-dependent ordering of substitution values can be produced and requires no complex ordering.

5.3 Application to Other Ciphers

While we demonstrated this methodology using the PRESENT-80 cipher, due to 16-bits of unused round keys during a given round, it is easily applicable to other ciphers and can require minimal effort for incorporation. Doing so would be simply dictated by cipher owner and extension of key length for incorporation of key-dependent permutation based networks is a simple and feasible solution, as demonstrated for existing ciphers, while an additional key-schedule for such key-dependent permutation and substitution based networks can easily be developed and implemented as well, further expanding the key-space, complexities, and securities involved in attacking such systems.

5.4 Use with other methods

While this method shows to increase the differential of affected bits via key-dependent structures, the PRESENT-80 cipher is also known to be susceptible to both fault-injection and side-channel analysis, with a proven countermeasure to both of these attacks presented in [6]. The application of this countermeasure is via SBox decomposition into shared registers using the system proposed in [12] and implementation of masking circuitry. As alterations we present to the PRESET-80 cipher round structure are minimal impact and only serve to further reduce leakage, in applicable cases, altering the round structure for use with existing methods can trivially be implemented.

6 Conclusion

As shown, the use of controlled key-dependent ICN primitives enhances resilience against differential-plaintext attacks while demonstrating, via PRESENT-80, that no impacts to performance were incurred and in the most optimal scenario system power was reduced. Additionally, the use of this design methodology is highly flexible, applicable to SPN based networks while easily integrated into existing cipher architectures and still able to be seamlessly used with other methods and countermeasures to further enhance overall cipher security.

References

1. Spartan-6 fpga configurable logic block user guide. URL https://www.xilinx.com/support/documentation/user_guides/ug384.pdf
2. Ara, T., Shah, P.G., M, P.: Dynamic key dependent s-box for symmetric encryption for iot devices. In: 2018 Second International Conference on Advances in Electronics, Computers and Communications (ICAECC), pp. 1–5 (2018). DOI 10.1109/ICAECC.2018.8479442

3. Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J., Seurin, Y., Vikkelsoe, C.: Present: An ultra-lightweight block cipher. Proceedings of the 9th International Workshop on Cryptographic Hardware and Embedded Systems pp. 450–466 (2007)
4. Breier, J., Jap, D., Bhasin, S.: Scadpa: Side-channel assisted differential-plaintext attack on bit permutation based ciphers. In: 2018 Design, Automation Test in Europe Conference Exhibition (DATE), pp. 1129–1134 (2018). DOI 10.23919/DATE.2018.8342180
5. bin Dai, Z., Xiang, N.: Fast bit permutation instruction based on omega+omega network. In: 2007 7th International Conference on ASIC, pp. 153–156 (2007). DOI 10.1109/ICASIC.2007.4415590
6. De Cnudde, T., Nikova, S.: Securing the present block cipher against combined side-channel analysis and fault attacks. IEEE Transactions on Very Large Scale Integration (VLSI) Systems **25**(12), 3291–3301 (2017). DOI 10.1109/TVLSI.2017.2713483
7. Guttman, B., Roback, E.A.: Sp 800-12. an introduction to computer security: The nist handbook (1995)
8. Lee, R.B., Shi, Z., Yang, X.: Efficient permutation instructions for fast software cryptography. IEEE Micro **21**(6), 56–69 (2001). DOI 10.1109/40.977759
9. Li, H., Gao, F.: Design and implementation of reconfigurable bit permutation system based on waksman network. In: 2010 Third International Conference on Information and Computing, vol. 2, pp. 113–116 (2010). DOI 10.1109/ICIC.2010.122
10. Nejad, F.H., Sabah, S., Jam, A.J.: Analysis of avalanche effect on advance encryption standard by using dynamic s-box depends on rounds keys. In: 2014 International Conference on Computational Science and Technology (ICCST), pp. 1–5 (2014). DOI 10.1109/ICCST.2014.7045184
11. Portz, M.: On the use of interconnection networks in cryptography. In: D.W. Davies (ed.) Advances in Cryptology — EUROCRYPT '91, pp. 302–315. Springer Berlin Heidelberg, Berlin, Heidelberg (1991)
12. Poschmann, A., Moradi, A., Khoo, K., Lim, C., Wang, H., Ling, S.: Side-channel resistant crypto for less than 2300ge. Journal of Cryptology **24**(2), 332–345 (2011)
13. Shannon, C.E.: Communication theory of secrecy systems
14. Stallings, W., Brown, L.: Computer Security: Principles and Practice, 1st edn. Prentice Hall Press, Upper Saddle River, NJ, USA (2008)
15. Yang, X., Lee, R.B.: Fast subword permutation instructions using omega and flip network stages. In: Proceedings 2000 International Conference on Computer Design, pp. 15–22 (2000). DOI 10.1109/ICCD.2000.878264
16. Young, C.P., Chia, C.C., Chen, L.B., Huang, I.J.: On-chip-network cryptosystem: A high throughput and high security architecture. In: APCCAS 2008 - 2008 IEEE Asia Pacific Conference on Circuits and Systems, pp. 1276–1279 (2008). DOI 10.1109/APCCAS.2008.4746260