AN ULTRA COMPACT BLOCK CIPHER FOR SERIALIZED ARCHITECTURE IMPLEMENTATIONS

Cheng Wang and Howard M. Heys

Electrical and Computer Engineering
Memorial University of Newfoundland
St. John's, Newfoundland, Canada
{cwang, howard}@engr.mun.ca

ABSTRACT

In this paper, we present a new block cipher, referred as PUFFIN2, that is designed to be used with applications requiring very low circuit area. PUFFIN2 is designed to be implemented exclusively with CMOS technologies and in a serialized architecture, so that the maximum reuse of hardware components is achieved resulting in a very compact implementation. PUFFIN2 has a block size of 64 bits and a key size of 80 bits. Compared with a serialized implementation of cipher PRESENT, which has the same block size and key size and is claimed as the smallest practical block cipher implementation to date, our cipher has 16% fewer gates using the same CMOS technology. Further, PUFFIN2 inherently supports both encryption and decryption while the serialized PRESENT is an encryption-only implementation.

Index Terms— cryptography, block cipher, ASIC, hardware implementation

1. INTRODUCTION

Lightweight block cipher design is one of the recent trends in symmetric key cryptography. This trend is motivated by the wide variety of applications of compact embedded devices (e.g. smart cards, RFID tags) and the huge market demand for them at low price and low power consumption. To cater to this trend, there has been quite a few proposals dedicated to lightweight block cipher design and implementation in recent years, and some of the representative examples are PRESENT [1][2], HIGHT [3], SEA [4], and PUFFIN [5]. Among the ASIC implementations of these lightweight block ciphers, PRESENT is the most compact encryption-only cipher and PUFFIN is the smallest one capable of both encryption and decryption operations.

It is well known that an efficient method to minimize hardware area is to reuse the single piece of a hardware component for multiple times instead of replicating identical pieces for simultaneous operation. This hardware reduction method, also known as a serialized architecture, is well suited to block ciphers which usually involve cryptographic components consisting of identical function blocks, e.g. non-linear substitution layers generally consist of identical S-boxes. Although this reduction of hardware area comes at the penalty of increased execution time, the compromised timing performance is still acceptable for many applications at which lightweight block ciphers are targeted. The serialized architecture is firstly exploited and applied to PRESENT in [2] and this is the smallest known implementation of a practical block cipher. In the reminder of the paper, this implementation is called serialized PRESENT.

PUFFIN2 is a block cipher named after its predecessor PUFFIN and designed to be implemented exclusively with a serialized architecture, and it is expected that PUFFIN2 is more efficient than PRESENT for hardware implementation with serialized architecture. The capability of both encryption and decryption is another design goal of PUFFIN2, and in the following it will be shown that the datapath of PUFFIN2 is exactly same for encryption and decryption, so there is no hardware overhead to accommodate the difference between encryption and decryption operations. PUFFIN2 has the same block size and key size as PRESENT, 64-bit and 80-bit respectively. Compared with the widely adopted 128-bit key size and 128-bit block size of AES [6], an 80-bit key size with a 64-bit block size can result in a compact implementation and still provide sufficient security for typical low cost smart devices. Based on our ASIC implementation experiments, PUFFIN2 is realized with 1083 gates which is 16% less than the serialized PRESENT with 1296 gates.

2. CIPHER SPECIFICATION

The proposed block cipher PUFFIN2 adopts a simple involutional substitution-permutation network (SPN) with a data block size of 64 bits and key size of 80 bits and consists of 34 rounds. The key schedule of the cipher generates 64-bit round-keys for each round on-the-fly (that is, in parallel to the processing of the cipher data). Encryption and

Table 1. S-box Mapping (in Hexadecimal) [5]

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

Table 2. 64-bit Permutation [5]
(input = row × 8 + column + 1)

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	13	2	60	50	51	27	10	36
1	25	7	32	61	1	49	47	19
2	34	53	16	22	57	20	48	41
3	9	52	6	31	62	30	28	11
4	37	17	58	8	33	44	46	59
5	24	55	63	38	56	39	15	23
6	14	4	5	26	18	54	42	45
7	21	35	40	3	12	29	43	64

decryption processes are identical so the same datapath can be used for both processes.

2.1. Basic components

Each round function of PUFFIN2 consists of 3 layers, a nonlinear substitution layer S, a key addition layer A and a permutation layer P. The nonlinear substitution layer S is composed of 16 identical 4x4 S-boxes, which are the same as the S-boxes used in PUFFIN and the S-box mapping is shown in Table 1. 4x4 S-boxes (which are small compared to the 8x8 S-boxes of AES) are often found in lightweight block ciphers because their implementations are compact and their comparative weakness in security strength can be compensated by an increased number of rounds. The key addition layer A performs a bitwise XOR with the 64-bit data block and the 64-bit round-key provided by the key schedule. The permutation layer P is a bit transposition of the 64-bit data block. The permutation scheme of PUFFIN2 is borrowed from the 64-bit data block permutation of PUFFIN which fulfills the criterion that no two outputs of a 4×4 S-box are connected to the same S-box in the next round. The permutation scheme is listed in Table 2. As can be seen from the tables, an important property that both the S-box mapping and the permutation possess is that they are involutions. A block cipher based on involutional function components can hold identical encryption and decryption processes, which is one of the design goals of PUFFIN2.

2.2. Encryption and decryption process

The encryption and decryption processes are shown in Figure 1, where *Kr* denotes the *r*-th round-key and *K'r=P(Kr)*. The whole process consists of 34 rounds plus an extra substitution layer. The explanation of selecting 34 as the number of rounds is given in the next sub-section. The extra substitution layer is required to form identical encryption and decryption processes. For each round of the encryption/decryption process, the 64-bit input data goes through the substitution layer S, the permutation layer P and then adds with the round key to generate the input of the next round.

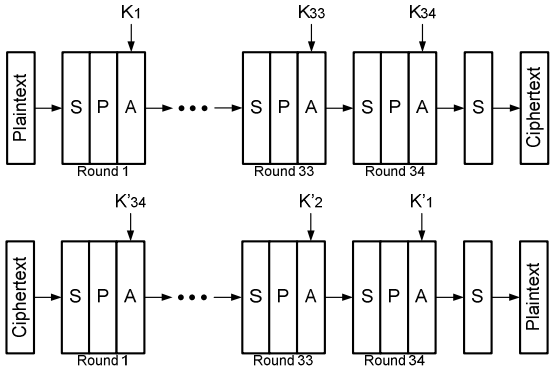

Figure 1. Block diagram of the encryption (top) and decryption (bottom) processes

According to Figure 1, the encryption process of PUFFIN2 can be represented as follows:

$$\alpha_{34}[K_1, K_2, ..., K_{34}] = O_{r=1}^{34}(S \circ P \circ A_{Kr}) \circ S \ .$$

In the above expressions, the notation, " ∘ ", means the concatenation of the basic operation in one stage such as substitution S and Permutation P. The notation "*O*" is used to represent the concatenation of 34 rounds of operation of $(S \circ P \circ A)$. Decryption should be as follows:

$$\begin{aligned}\alpha_{34}^{-1}[K_1, K_2, ..., K_{34}] &= S \circ O_{r=34}^{1}(A_{Kr} \circ P \circ S) \\ &= O_{r=34}^{1}(S \circ A_{Kr} \circ P) \circ S \ .\end{aligned}$$

Because the substitution layer S and the permutation layer P are involutional, we can have the following relationship:

$$A_{Kr} \circ P \equiv P \circ A_{P(Kr)} \ ,$$

and therefore, we can obtain the following:

$$\alpha_{34}^{-1}[K_1, K_2, ..., K_{34}] = O_{r=34}^{1}(S \circ P \circ A_{P(Kr)}) \circ S \ .$$

The above expression is consistent with the decryption process shown in Figure 1, which means the decryption process is similar in form to the encryption process. In the decryption process, the round-keys used in encryption are permuted with P and applied in the reverse order.

2.3. Key schedule

The key schedule of PUFFIN2 operates on an 80-bit key and generates a 64-bit key for each round on-the-fly. The components used by the key schedule are listed in Table 3 and the key schedule is demonstrated in Figure 2. The key schedule consists of 34 round functions plus an extra substitution layer at the beginning. Each of the round function is comprised of a permutation layer PL64 or PR64

Table 3. Description of the components of the key schedule

Component	Function
S80	Substitution of the 80 bits
PL64	Permutation of the left 64 bits
PR64	Permutation of the right 64 bits
L64	Selection of the left 64 bits
R64	Selection of the right 64 bits

Table 4. Round distribution of PL64, PR64, L64 and R64

Round #	Permutation	Selection
$r = 1, 2, 33, 34$	PL64	L64
$r = 3, 4, 31, 32$	PR64	R64
$r = 5 + 4n, 0 \le n \le 6$	PR64	R64
$r = 6 + 4n, 0 \le n \le 6$	PR64	R64
$r = 7 + 4n, 0 \le n \le 5$	PL64	L64
$r = 8 + 4n, 0 \le n \le 5$	PL64	L64

and a substitution layer S80. PL64 or PR64 permutes the left or right 64 bits of the 80-bit round input and then S80 performs the substitution on the 80 bits of key data. Depending on the selection component L64 or R64, each 64-bit round key is generated by taking the left or right 64 bits of the 80-bit intermediate value that feeds to the corresponding round function. The detailed distribution of PL64 and PR64 along with L64 and R64 is shown in Table 4. The irregular distribution of PL64 and PR64 for each round is intended to prevent related-key attacks, and will be discussed in Section 3.

In order to maximize hardware resource reuse to achieve a compact implementation, the substitution component S80 is designed to consist of 4x4 S-boxes that are same as the S-box used in the encryption and decryption processes, and the 64-bit permutation mapping in PL64 and PR64 is also the same mapping as in the permutation layer in the encryption and decryption process.

The key schedule of PUFFIN2 is designed to be involutional to fulfill the design goal of a full involutional block cipher. The involutional property is achieved through the following measures. In the first place, all basic components in the key schedule are involutional. Secondly, the distribution of PL64 PR64 along with L64 and P64 is symmetric, and in order to achieve this, the round number of the key schedule has to be a number that is double of an odd number and consequently 34 is selected as the round number of the key schedule as well as the encryption and decryption processes. As will be noted in Section 3, 34 rounds are also adequate to achieve an appropriate level of security. Thirdly, there is an extra substitution layer S80 at the beginning of the key schedule which makes the forward path of round key generation identical to its backward path. The PL64 and S80 in the last round (round 34) of the key schedule are only useful to compute the decryption key corresponding to an encryption key. By applying a decryption key to the key schedule, the round keys would be generated in the reverse order of that made by the encryption key. It is also necessary

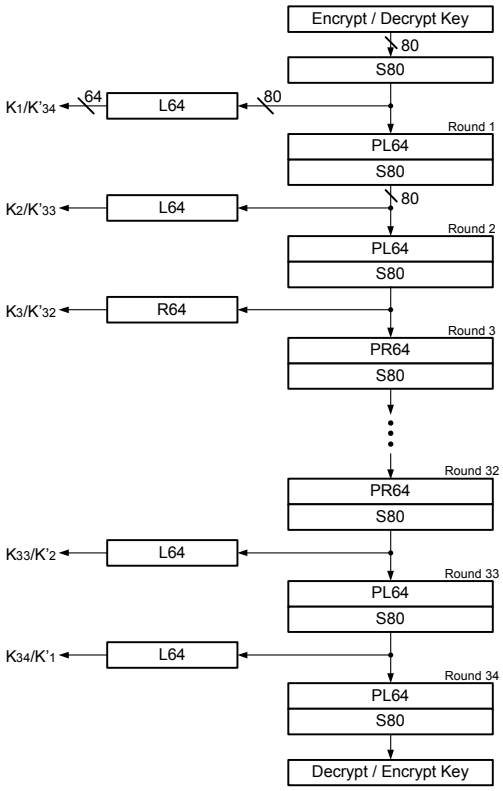

Figure 2. Block diagram of the key schedule

to note that the round keys generated for decryption are permuted version of the round keys used in encryption, and this feature is required to provide the correct decryption round keys for the decryption process mentioned in the last sub-section.

3. SECURITY ANALYSIS

In this section, we analyze the security strength of PUFFIN2 under differential and linear cryptanalysis and two major key schedule attacks.

3.1. Differential and linear cryptanalysis

Our proposed block cipher PUFFIN2 shares the same S-box and permutation mapping in encryption and decryption process as PUFFIN, so the differential and linear cryptanalysis results of PUFFIN2 can be easily derived from that of PUFFIN in [5].

For differential cryptanalysis [7], the maximum differential characteristic probability of the S-box is $p_\delta = 1/4$, and based on the 4x4 S-boxes and the involutional permutation each round has at least one active S-box to form the path for a differential characteristic. Hence, the upper bound of the differential probability over 32 rounds is given by:

$$p_\Omega \le p_\delta^{32} = 2^{-64} .$$

The differential probability p_Ω indicates that about 2^{64} plaintexts would be required to mount a successful attack, the complexity of which is close to a brute force attack on a 64-bit cipher. Therefore, it is reasonable to consider PUFFIN2 to be enough resistant to differential cryptanalysis.

For linear cryptanalysis [8], the maximum linear approximation probability bias of the S-box is $|\varepsilon_S| = 1/4$. Similar to the case in differential cryptanalysis, there is at least one active S-box involved in each round to form a linear approximation. Hence, the upper bound of the linear approximation bias ε_L of 32 rounds is calculated with the piling-up lemma [8] as follows:

$$|\varepsilon_L| \le 2^{31} |\varepsilon_S|^{32} = 2^{-33}.$$

According to [8], the number of the known plaintexts required to perform linear cryptanalysis is proportional to $1/\varepsilon_L^2$, which means an effective attack on PUFFIN2 with linear cryptanalysis requires about 2^{66} known plaintexts and therefore is considered to be an impractical attack.

3.2. Related-key attacks

The related-key attack was proposed in [9]. It exploits the regularity of the relationships between key schedule rounds and uses the chosen key relations to retrieve the secret key information. The related-key attack generally finds its application on those block ciphers that use the same algorithm to generate round keys for all the rounds. It is easy to see that our block cipher does not have this regularity property in the key schedule because the permutation layers PL64 and PR64 are not regularly distributed among rounds, and hence it is resistant to the related-key attack.

3.3. Weak keys

Weak keys [10] are keys that make the key schedule produce identical round keys for all or some of the rounds. For the key schedule of PUFFIN2, due to the existence of non-linear substitution layers, we do not find any weak key in the key space.

4. SERIALIZED ARCHITECTURE FOR HARDWARE IMPLEMENTATION

The proposed block cipher PUFFIN2 is designed to be efficiently implemented with a serialized hardware architecture. A serialized architecture is the architecture where multiple identical hardware components that work for multiple tasks simultaneously are mapped to one piece of the hardware component that works for the multiple tasks in series. Generally, a serialized architecture can lead to the minimal hardware implementation among a variety of implementation architectures. In this section, we introduce a serialized architecture for which the proposed block cipher PUFFIN2 is well suited in and that results in an ultra compact implementation.

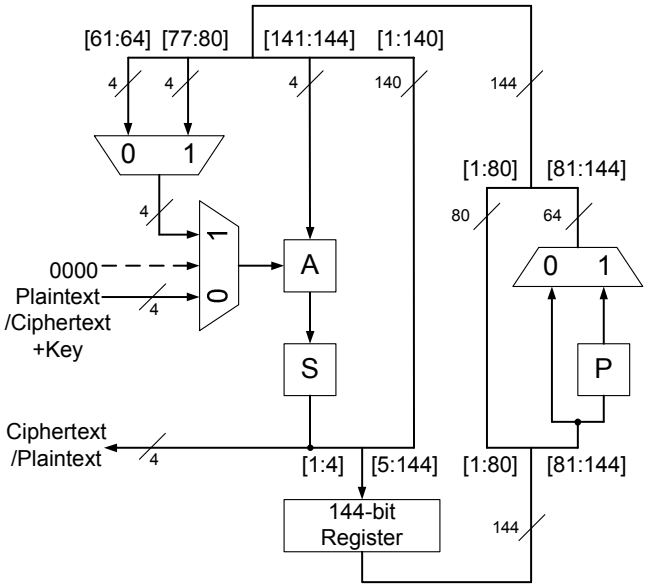

Figure 3. Serialized architecture of PUFFIN2

The serialized architecture for PUFFIN2 is shown in Figure 3. This architecture is constructed with two 4-bit 2-to-1 multiplexers, a 64-bit 2-to-1 multiplexer, a 4-bit XOR adder "A", a 4x4 S-box "S", a 64-bit permutation "P", and a 144-bit register. The architecture conceives a 144-bit wide datapath and the hardware components operate on the datapath.

It is also worthy to mention that there is a 4-bit rotation structure on the datapath, which is crucial to ensure the hardware resources are shared properly in series and the block cipher algorithm is running correctly in the architecture. The 4-bit rotation structure is realized by crossover wiring that maps bits 1 to 140 to bits 5 to 144 and bits 141 to 144 to bits 1 to 4 (through the adder "A" and the S-box "S").

The 4-bit 2-to-1 multiplexer with a dashed 4-bit zero input in Figure 3 is called the first 4-bit multiplexer and the other one is called the second 4-bit multiplexer. The first 4-bit multiplexer is capable to output a zero vector independent of its two inputs, and this is achieved by ANDing the 4-bit output of the multiplexer with a signal bit from the controller.

In the following, we describe the work flow of the serialized architecture with the example of plaintext and key loading procedure and the first round of the encryption process. We call the bits that carry plaintext information and key information during the encryption process as internal plaintext bits and internal key bits, respectively. A 64-bit plaintext plus an 80-bit encryption key is loaded in units of 4 bits through the first 4-bit multiplexer to the datapath. The first 4-bit unit is presented to the input of the 144-bit register in the 1st clock cycle and becomes available at the output of

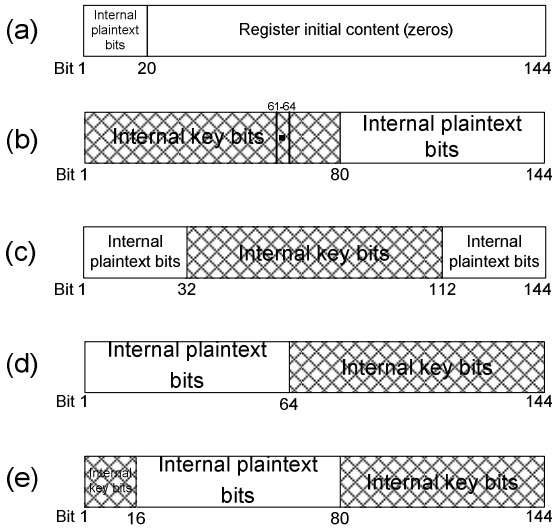

Figure 4. Contents of the 144-bit register at clock cycles 6, 37, 45, 53 and 57

the register in the 2nd clock cycle. Each subsequent 4-bit unit is added with a 4-bit zero vector and then fed through the S-box before being stored in the 144-bit register. The 4-bit zero vector is generated by the initial output of the 144-bit register. The 4-bit rotation structure makes sure each 4-bit unit is added with a 4-bit zero vector and stored in the register bits next to the last 4-bit unit. The loading procedure of the plaintext plus the key takes 36 clock cycles and during this period the first substitution layers in the encryption process and key schedule are also performed. In the 37th clock cycle, the 64-bit internal plaintext bits are permuted with the 64-bit permutation by selecting position 1 of the 64-bit 2-to-1 multiplexer, and then the right most 4 bits of the updated internal plaintext bits are added with the right most 4 bits of the left 64 bits of the internal key bits by selecting position 0 of the second 4-bit multiplexer. It takes 16 clock cycles to complete the key addition of 64 bits and ends at the 52nd clock cycle. In the 53rd cycle, the right-most 4 bits of the 80-bit internal key bits are added with a 4-bit zero vector in A and then substituted by the S-box. In the 57th cycle, the left 64 bits of the 80-bit internal key bits are permuted with the 64-bit permutation. In order to better demonstrate the work of the architecture, the contents of the 144-bit register at clock cycles 6, 37, 45, 53 and 57 are shown in Figure 4 (a), (b), (c), (d) and (e), respectively. The dotted 4-bit block in Figure 4 (b) is the 4 bits used to be added with the right most 4 bits of the internal plaintext (after the permutation) in the 37th cycle.

The 64-bit internal plaintext bits and the 80-bit internal key bits are rotated within the 144-bit register and it takes 36 cycles to complete a full rotation. The period of 36 cycles is also the time to complete a round of the encryption/decryption process. Hence, the total time to complete the entire encryption/decryption including the 16 cycles for the initial loading of the plaintext is given by:

Table 5. Implementation results of PUFFIN2 and serialized PRESENT

	Area (GEs)	Maximum Frequency	Cycles	Throughput @100KHz
PUFFIN2	1083	326.8MHz	1240	5.2Kbps
Serialized PRESENT	1296	346.0MHz	563	11.4Kbps

$$(16+36\times34)cc=1240cc\,,$$

where *cc* represents one clock cycle.

5. HARDWARE IMPLEMENTATION RESULTS

The block cipher PUFFIN2 with the serialized architecture has been implemented and synthesized with the 0.18-µm CMOS standard cell library from TSMC available through CMC Microsystems [11]. Synopsys Design Compiler version X-2005.09 has been used as our synthesis tool. We also implemented the serialized PRESENT from [2] which is claimed as the smallest implementation of a block cipher with 64-bit block size. Both of the implementations are datapath-only implementations, which means their controllers are not included in the implementations, and in both cases the controllers are negligible because they can be realized with a small counter and a small amount of combinational logic. Our implementation results of PUFFIN2 and the serialized PRESENT are shown in Table 5. In the table, the metric of gate equivalents (GEs) is used, where a unit of 1 GE represents an area equivalent to a 2-input NAND gate.

According to Table 5, the implementation of PUFFIN2 is 16% smaller than the serialized PRESENT implementation. As a trade-off PUFFIN2 takes almost double the time of the serialized PRESENT to process the same amount of data. In most lightweight applications, a large running time is not a serious issue.

It is necessary to point out that the gate count of the serialized PRESENT implementation claimed in [2] is 1075 GE. The 221 GE overhead of our implementation of the serialized PRESENT could be caused by the different synthesis library and the use of scan flip flops with integrated multiplexers in [2] instead of the normal flip flops and separated multiplexers found in our implementation. The same area reduction effect can be achieved in our implementation of PUFFIN2 with scan flip flops as long as the 144-bit register is moved to the output of the 64-bit 2-to-1 multiplexer to form the integrated flip flops and multiplexers. The position of the 144-bit register is flexible in the serialized architecture, so this change would not have any influence on the functionality.

In order to have a clear comparison between the hardware complexity of PUFFIN2 and the serialized PRESENT, we list the count of the hardware components required for both implementations in Table 6. The 144-bit register in PUFFIN2 is divided into a 64-bit register and

Table 6. Count of hardware components of PUFFIN2 and serialized PRESENT

Components	PUFFIN2	Serialized PRESENT
64-bit register (384 GE)	1 (35.5%)	1 (29.6%)
80-bit register (480 GE)	1 (44.3%)	1 (37.0%)
64-bit 2-to-1 multiplexer (153 GE)	1 (14.1%)	1 (11.8%)
80-bit 2-to-1 multiplexer (192 GE)	0	1 (14.8%)
4-bit 2-to-1 multiplexer (10 GE)	2 (1.8%)	3 (2.3%)
4x4 S-box* (30 GE / 32GE)	1 (2.8%)	1 (2.5%)
4-bit XOR adder (11 GE)	1 (1.0%)	1 (0.9%)
5-bit XOR adder (14 GE)	0	1 (1.1%)
4 2-input AND gates (5 GE)	1 (0.5%)	0
Total gate count	1083 GE (100%)	1296 GE (100%)

* PUFFIN2 and the serialized PRESENT use different S-boxes with slightly different area.

an 80-bit register in Table 6, and the 36 4-bit 2-to-1 multiplexers in the two shift registers of the serialized PRESENT are merged and shown as a 64-bit 2-to-1 multiplexer and an 80-bit 2-to-1 multiplexer in Table 6.

From Table 6, we can see the major area difference between PUFFIN2 and the serialized PRESENT comes from the 80-bit 2-to-1 multiplexer, which accounts for 14.8% of the total area of the serialized PRESENT and does not exist in PUFFIN2. It is also noticeable in Table 6 that the 144-bit register takes 80% of the hardware resource of PUFFIN2, and this fact allows us to believe that the serialized implementation of PUFFIN2 has approached the area limit of the block ciphers that have similar block size and key size.

6. CONCLUSION

In this paper we have proposed a new block cipher PUFFIN2 based on an involutional SPN structure. The cipher with a 64-bit block size and an 80-bit key size can provide sufficient security for low cost embedded devices and support both encryption and decryption. We also introduced a serialized architecture based on which PUFFIN2 can be implemented with an ultra compact size. Compared with the serialized PRESENT implementation, the datapath of PUFFIN2 uses 16% fewer gates. In general, the PUFFIN2 block cipher is a secure, area-efficient structure in comparison to other proposed compact block ciphers.

7. ACKNOWLEDGEMENTS

This work was funded by the Natural Sciences and Engineering Research Council of Canada (NSERC) and facilitated by tools provided by CMC Microsystems.

8. REFERENCES

[1] A. Bogdanov, L.R. Knudsen, G. Leander, C. Paar, A. Poschmann, M.J.B. Robshaw, Y. Seurin, and C. Vikkelsoe, "PRESENT: An Ultra-Lightweight Block Cipher," *Cryptographic Hardware and Embedded Systems (CHES 2007)*, Springer-Verlag, LNCS 4727, pp. 450-466, 2007.

[2] C. Rolfes, A. Poschmann, G. Leander, and C. Paar, "Ultra-Lightweight Implementations for Smart Devices – Security for 1000 Gate Equivalents," *Smart Card Research and Advanced Application Conference (CARDIS 2008)*, Springer-Verlag, LNCS 5189, pp. 89–103, 2008.

[3] D. Hong, et al., "HIGHT: A New Block Cipher Suitable for Low Resource Device," *Cryptographic Hardware and Embedded Systems (CHES 2006)*, Spring-Verlag, LNCS 4249, pp. 46-59, 2006.

[4] F.-X. Standaert, G. Piret, N. Gershenfeld, and J.-J. Quisquater, "SEA: A Scalable Encryption Algorithm for Small Embedded Applications," *Smart Card Research and Applications (CARDIS 2006)*, Springer-Verlag, LNCS 3928, pp. 222–236, 2006.

[5] H. Cheng, H.M. Heys, C. Wang, "PUFFIN: A Novel Compact Block Cipher Targeted to Embedded Digital Systems," *11th Euromicro Conference on Digital System Design Architectures, Methods and Tools (DSD 2008),* 2008.

[6] National Institute of Standards and Technology (NIST), "Advanced Encryption Standard (AES)," *Federal Information Processing Standard (FIPS) 197*, Nov. 2001.

[7] E. Biham and A. Shamir, "Differential Cryptanalysis of DES-like Cryptosystems," *Advances in Cryptology: CRYPTO'90*, Springer-Verlag, LNCS 537, pp. 2-21, 1991.

[8] M. Matsui, "Linear Cryptanalysis Method for DES Cipher," *Advances in Cryptology: EUROCRYPT '93*, Springer-Verlag, LNCS 765, pp. 386-397, 1994.

[9] E. Biham, "New Type of Cryptanalysis Attacks Using Related Keys," *Advances in Cryptology: EUROCRYPT '93*, Springer-Verlag, LNCS 765, pp. 229-246, 1994.

[10] J. H. Moore, and G. J. Simmons, "Cycle Structure of the DES for Keys Having Palindromic (or Antipalindromic) Sequences of Round Keys," *IEEE Transactions on Software Engineering*, vol. SE-13, no. 2, pp. 262-273, 1987.

[11] CMC Microsystems, www.cmc.ca