PUFFIN: A Novel Compact Block Cipher Targeted to Embedded Digital Systems

Huiju Cheng, Howard M. Heys, and Cheng Wang

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

Abstract

In this paper, we examine the digital hardware design and implementation of a novel compact block cipher, referred to as PUFFIN, that is suitable for embedded applications. An implementation of PUFFIN targeted to ASIC technology is considered. The proposed block cipher is designed to have a 64-bit block size, a 128-bit key, and is capable of both encryption and decryption operations. The cipher structure is based on the following features: a simple encryption process composed of permutations and substitutions based on 4×4 S-boxes, an identical datapath for both encryption and decryption facilitated by involutional operations, and a straightforward on-the-fly subkey generation composed of only a permutation and bit inversions. PUFFIN is found to perform well for implementations based on 0.18-micron CMOS technology. In comparison to other lightweight ciphers, PUFFIN has preferred features, low hardware complexity, and good throughput.

1. Introduction

In this paper, we propose a novel block cipher, called PUFFIN, based on a substitution-permutation network (SPN) structure [1]. It is designed for applications requiring low circuit area and is suitable for ASIC and FPGA implementations. The new cipher features a simplicity of design and, as an involutional block cipher, is easily implemented to be capable of both encryption and decryption functionality with a data block size of 64 bits and a key size of 128 bits. To achieve this we have utilized low complexity 4×4 S-boxes, instead of the expensive 8×8 S-boxes (which are found, for example, in the Advanced Encryption Standard (AES) [2]), as our nonlinear substitution components and they can be easily implemented in hardware with simple combinational logic of 4-bit Boolean functions. In addition, the encryption or decryption process may share the same hardware due to the involutional nature of the components in the cipher. We have also applied a very simple key schedule with only a permutation and bit inversions so that the subkeys can be derived on-the-fly and, hence, there is no need to store all the subkeys. This simple key schedule also ensures that the secret key can change in a clock cycle and, hence, the cipher is highly key-agile.

All these characteristics make the new cipher very efficient for hardware implementations which are targeted to low cost embedded applications. The cipher is also resistant to the two important classes of cryptanalysis: differential and linear cryptanalysis. Further, it also provides resistance to related-key attacks and does not have any weak keys in the total keyspace.

2. Background

In recent years, several papers have examined the digital hardware implementation of lightweight block ciphers targeted to embedded applications like smartcards and RFID tags [3]. For example, it is well known that the 8×8 S-box of AES is the greatest consumer of circuit area in a CMOS design and, as a result, in [4], a compact ASIC implementation of AES is presented, which achieves low hardware complexity through the re-use of a single component S-box circuit. This implementation requires only 3400 gates but is slow.

ICEBERG [5] is a proposal, with a 64 bit block size and an 128 bit key, that is intended for efficient, high speed applications targeted to reconfigurable hardware, but that is also suitable for compact applications. It is based on the SPN structure and uses S-boxes and permutations that are involutions. Although it supports both encryption and decryption operations, it does not appear to be as compact as other proposed ciphers.

DESL [6] is a lightweight variant of the Data Encryption Standard (DES) which makes use of only one S-box mapping and can therefore be made to more compact that DES (which uses 8 S-box mappings). DESXL is a strengthened DESL variant with a key size of 184 bits (although the effective key size is about 118 bits [3]).

Similar to our new compact block cipher PUFFIN, a

lightweight block cipher PRESENT [7] is a recently proposed SPN. PRESENT is a 64-bit block cipher with a key length of 80 or 128 and consists of 31 rounds. The substitution layer applies sixteen 4×4 S-boxes, but neither the S-box nor the 64-bit permutation is involutional, and unlike PUFFIN, PRESENT is designed and implemented to support encryption only. Hence, while achieving a very compact implementation of less than 2000 gates, PRESENT is not capable of supporting modes (eg. cipher block chaining) that require decryption. As well, the most compact implementation of PRESENT has only an 80 bit key size and is therefore only suitable to environments that can accept limited security.

Other proposed compact block ciphers such as mCrypton [8] and Hight [9] will be included in the discussion of results in Section 5.2.

The block cipher proposed in this paper is very compact, at least comparable in area complexity to other proposals for embedded block ciphers and, contrary to many other cipher implementations, is capable of encryption and decryption. In addition, PUFFIN has a large key size of 128 bits and, therefore, is suitable for a range of embedded applications, including those requiring a high level of security and those which make use of modes requiring decryption.

3. Specifications of PUFFIN

The new block cipher PUFFIN proposed in this paper applies a simple involutional SPN structure with a data block size of 64 bits and the key size is specified to be 128 bits. Although for some applications, a larger block size offers better security, for embedded applications, compact block ciphers are often proposed with a block size of 64 bits (eg. ICEBERG, DESXL, PRESENT). For the key size, generally 80 bits is considered a minimal requirement for low-security embedded applications. However, in practice a 128-bit key is able to provide adequate security for any application. (For example, AES has no specification for key sizes less than 128 bits.) Hence, for our cipher, we have assumed that a 128-bit key size is desired.

Generally, SPN ciphers require different datapaths for encryption and decryption because the inverse operations used in the decryption round are usually different from those forward operations applied in the encryption round. The advantage of our cipher is that all the components are involutional which means the inverse operations used in the decryption process can be the same as those in the encryption process. Hence, the involutional SPN structure allows a very efficient implementation to use identical hardware for both encryption and decryption. Other involutional ciphers have been previously proposed in other contexts [10][11], including, of course, ICEBERG [5].

3.1. Basic Components

In each round function of PUFFIN, three stages of operations are applied. The first stage is the nonlinear substitution layer, γ, which is composed of sixteen identical 4×4 S-boxes. Often, 8×8 S-boxes are used in block ciphers in consideration of their better nonlinear and differential properties. In our proposed new block cipher PUFFIN, we apply 4×4 S-boxes which are much more compact and have a lower critical path delay from input to output. Due to the use of more simple S-boxes, we need to increase the number of rounds required to produce the ciphertext in order to guarantee the security of the cipher against cryptographic attacks such as linear and differential cryptanalysis. The 4×4 S-boxes used in our cipher are the same as the S_0 mapping applied in ICEBERG [5] and the S-box mapping is shown in Table 1. From this table, we can see that the S-box is involutional.

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

input	0	1	2	3	4	5	6	7
output	D	7	3	2	9	A	C	1
input	8	9	A	B	C	D	E	F
output	F	4	5	E	6	0	B	8

In ICEBERG, 8×8 S-boxes are constructed from three layers of 4×4 S-boxes combined with two layers of eight 8-bit permutations. PUFFIN with just one layer of sixteen 4×4 S-boxes, not surprisingly, results in a more compact architecture than ICEBERG.

The second stage of the new block cipher's round is the key addition layer, σ, which is composed of the bitwise XOR between the 64-bit data block and the 64-bit subkey. The subkey used in each round of the encryption/decryption process can be derived from the secret key of the cipher by the key schedule.

The third stage of the round is the permutation layer, $P64$, which performs the transposition of the 64-bit data block in PUFFIN. The permutation can be implemented in wire crossings which do not cost any hardware gates. The permutation table for $P64$ is listed in Table A1 of the Appendix. It can be seen that permutation $P64$ in an involution and satisfies the property that no two outputs of a 4×4 S-box are connected to the same S-box in the next round.

An important requirement for an SPN (or, indeed, any block cipher) is the property of completeness [12]. Completeness is achieved in a cipher if all the ciphertext bits are dependent on all the plaintext bits. For PUFFIN, it can be shown that the completeness property is satisfied after five rounds when applying the S-boxes and permutations we have chosen. For the whole encryption process of our new cipher, 32 rounds

are adequate to provide the necessary security of the cipher. This will be justified in Section 4.

3.2. Encryption and Decryption Process

Each round function of the encryption or decryption process is composed of three stages: substitution γ, key addition σ, and permutation $P64$. In our new cipher, 32 rounds are needed to securely produce the ciphertext. For the encryption, the 64-bit plaintext is first added with the secret key and then permuted using $P64$. Following is 32 identical round functions with the three stages of substitution, round key addition, and permutation. Figure 1 shows the diagram of the encryption process.

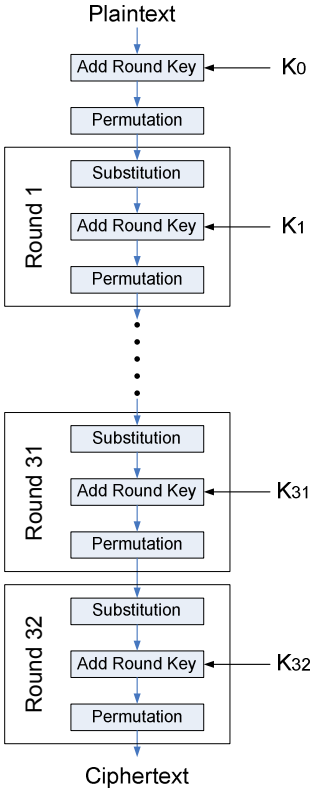

Figure 1. Block Diagram of the Encryption Process

The encryption process for PUFFIN can be represented as follows:

$$\alpha_{32}[K_0, K_1, ..., K_{32}] = \sigma_{K_0} \circ P64 \circ O_{r=1}^{32}(\gamma \circ \sigma_{K_r} \circ P64).$$

In the above expressions of the encryption process, the notation, "$\circ$", means the concatenation of the basic operation in one stage such as substitution γ and key addition σ_{Kr}. The notation "O" is used to represent the concatenation of 32 rounds of operation of ($\gamma \circ \sigma_{Kr} \circ P64$). There are 33 cipher subkeys, K_0 to K_{32}.

Our compact new cipher is an involutional cipher so that the decryption process is identical to the encryption process after an additional permutation is performed on the subkeys used in the decryption process. However, the subkeys in the decryption process still should be used in a reverse order.

Consider the following relationship:

$$P64 \circ \sigma_{K_r} \equiv \sigma_{P64(K_r)} \circ P64 .$$

The decryption process of PUFFIN can be obtained as follows:

$$\begin{aligned} \alpha_{32}^{-1}[K_{32}, K_{31}, ..., K_1, K_0] &= O_{r=32}^{1}(P64 \circ \sigma_{K_r} \circ \gamma) \circ P64 \circ \sigma_{K_0} \\ &= P64 \circ \sigma_{K_{32}} \circ O_{r=31}^{0}(\gamma \circ P64 \circ \sigma_{K_r}) . \end{aligned}$$

Hence, we can obtain:

$$\begin{aligned} &\alpha_{32}^{-1}[K_{32}, K_{31}, ..., K_1, K_0] \\ &= \sigma_{P64(K_{32})} \circ P64 \circ O_{r=31}^{0}(\gamma \circ \sigma_{P64(K_r)} \circ P64) . \end{aligned}$$

As the representation shows above, the decryption process can be performed in the same way as encryption except that the subkeys should be used in a reverse order and a corresponding permutation $P64$ is applied to them before they are used for the key addition stage in decryption.

3.3. Key Schedule

The key schedule portion of the cipher is responsible for generating the 33 subkeys from the 128-bit cipher key. We have designed a very simple key schedule to be applied in our new compact block cipher PUFFIN. The key schedule is designed to provide resistance to key schedule cryptanalysis. All the subkeys are generated on-the-fly in both the encryption and decryption process. Therefore, it is not necessary to store all the subkeys. For the purpose of compactness, our simple key round function only performs a permutation and selected bit inversions without any nonlinear substitution operation. The permutations are implemented as wirings which do not require any hardware gates and the bit inversions also cost very limited hardware.

The key schedule operates on 128 bits, initialized by the cipher key using permutation and selected bit inversions. The 128-bit permutation, used in the key schedule and listed in Table A2 of the Appendix A, performs a specially selected transposition of the 128 bits of the input key bits, and after this permutation, the selected four bits in position 1, 2, 3, and 5 will be inverted. As a result, four different positions of the bits in the original cipher key are inverted after each key round operation. The design goal of this combination of permutation and inversion is to allow the key schedule to be free of the weak keys which would cause security problems in the cipher [13].

In addition, in order to be resistant to the related-key attack [14], the selected four bit inversions are not processed in each key round function so that

non-regularity can be provided between the generation algorithms of the subkeys. In our key schedule, the second, fifth, sixth and eighth key rounds will not perform the inversions, while the remaining 28 rounds will. We will discuss the security analysis of the key schedule in more detail in Section 4.

The 128-bit permutation applied in the key schedule is not designed to be involutional. Therefore, the hardware of the key schedule used in the encryption and decryption can not be the same. The corresponding inverse 128-bit permutation is used in the subkey generation for decryption. In the key round function of decryption, the corresponding selected four bits in position 30, 62, 71 and 120 will be inverted after the 128-bit permutation.

For the encryption or decryption of PUFFIN, the subkeys require only 64 bits and, hence, an additional key selection is needed in the key schedule to choose 64 bits out of the 128 key bits generated in the key schedule. To achieve the goal of a very efficient hardware implementation of a compact cipher, we prefer to apply a simple key selection function without costing any hardware resource. Another design criterion related to security is that the fewest number of rounds should be required to have every bit of the 128-bit key chosen at least once for a subkey.

We have tried many simple ways to perform the key selection and have calculated the percentage of the bits being used in PUFFIN encryption or decryption among the whole 128 key bits after each round. Table 2 shows the number of the key bits being used after each round function until all the 128 key bits have been applied in the encryption or decryption at least once based on different key selection operations. For example, if we choose the even bits from the 128-bit key after each key round function the whole 128 bits of the secret key would be chosen at least once after six rounds. In our design, with the purpose of reaching the goal that all 128 key bits being used in the encryption or decryption within four key round functions, we have found a preferred selection of 64 bits out of the 128-bit key by examining 100,000 randomly chosen selections for the 64-bits to be applied in a round. This preferred selection function table used in the encryption/decryption process is listed in Table A3 of the Appendix.

Table 2. Number of Key Bits Being Used Based on Different Key Selections

Rnd	1	2	3	4	5	6	7	8
Even Bits	64	99	118	126	127	128		
Right half	64	100	117	121	123	125	127	128
Pref'd	64	97	117	128				

4. Security Analysis

In this section, we will discuss the resistance provided by our new block cipher against two important classes of cryptanalysis: differential and linear cryptanalysis. It will also be shown that our new cipher is resistant to two major key schedule insecurities: related-key attacks and weak keys.

4.1. Differential Cryptanalysis

Differential cryptanalysis [15] is a chosen plaintext attack that exploits the high probability of particular plaintext pair XOR differences and the corresponding differences of the ciphertexts. The differential characteristic probability determines the complexity of the attack and an upper bound can be estimated by using the minimum number of S-boxes involved in the attack over the rounds of the cipher and the highest differential characteristic probability of each S-box involved. The input and output XOR pair differences of involved S-boxes are combined from round to round in the way that the nonzero output pair differences are used as the input pair differences of the S-box in the next round. As a result, an (R-1)-round differential characteristic probability of the plaintext differences and the differences of the input to the last round can be achieved, where R represents the number of rounds in the cipher.

For the 4×4 S-box applied in our cipher, the maximum differential characteristic probability of the S-box can be determined to be $p_\delta = 1/4$. To calculate the upper bound of the complete differential characteristic probability of the cipher, we need the maximum characteristic probability of the S-box in each round and the fewest active S-boxes involved in all rounds of the cipher. Based on the 4×4 S-boxes and the involutional permutation between the S-boxes, it is possible that a characteristic probability exists with only one active S-box affected in each round. As a result, the upper bound of the probability of the (R-1)-round differential characteristic consisting of the plaintext XOR difference and the input XOR difference to the last round of the cipher can be represented as follows:

$$p_\Omega \le p_\delta^{R-1} = 2^{-62}$$

where p_δ represents the maximum differential characteristic of the S-box which is 1/4 as previously noted, and R represents the number of cipher rounds which is defined as 32 in our cipher. Consequently, since the complexity of the attack is inversely related to the differential probability p_Ω, 2^{62} chosen plaintext pairs would be needed to attack the cipher. This approaches the complexity required in a dictionary attack on a 64-bit cipher and hence it is reasonable to

interpret the cipher as being resistant to differential cryptanalysis. Our computation of the differential characteristic probability is under the assumption that the difference XOR pairs of the S-boxes are independent which is a typical assumption made in security analyses.

4.2. Linear Cryptanalysis

Linear cryptanalysis [16] is a known plaintext attack undertaken by constructing a linear path which uses a probabilistic relationship between the input and output bits of each S-box, combining the linear path of the S-box from round to round, and finally obtaining a high probability linear approximation expression between the plaintext, ciphertext and key without any intermediate values. The key information of the cipher may be extracted by using this linear approximation expression. Since the only nonlinear component of the cipher is the S-box, the linear approximation between the inputs and outputs of the S-boxes are important in constructing a (R-1)-round linear approximation based on the algorithm 2 presented in [16]. The linear approximation of the S-boxes can be concatenated to construct a linear expression with probability bias away from 1/2 involving only plaintext and the second last round outputs without any intermediate bits.

Under the assumption that each S-box approximation is independent, the piling-up lemma in [16] can be used to determine the upper bound of the (R-1) linear approximation probability bias. To calculate the upper bound, we need the maximum linear approximation probability bias of the S-box in each round to be applied and the fewest active S-boxes involved in the whole rounds of the cipher. Based on the 4×4 S-box and the involutional permutation between the S-boxes, it is possible that a high linear approximation probability bias exists with only one active S-box affected in each round. As a result, the upper bound of the bias, ε_L, of the (R-1) linear approximation probability where the linear expression consists of the plaintext data bits and the data bits of the second last round output can be represented as follows:

$$|\varepsilon_L| \le 2^{R-2} |\varepsilon_S|^{R-1} = 2^{-32}$$

where $|\varepsilon_S|$ represents the maximum linear approximation probability bias of the S-box which is 1/4 for the S-box of PUFFIN, and the number of rounds in PUFFIN is R = 32. In [16], Matsui shows that the number of the known plaintexts needed to perform linear cryptanalysis is proportional to $1/\varepsilon_L^2$. Therefore, about 2^{64} known plaintexts would be needed to attack our compact new cipher and linear cryptanalysis is not a practical attack against PUFFIN. Our analysis is based on the assumption, typically used in security analyses, that the S-box approximations used in the overall linear approximation may be treated independently.

4.3. Related-Key Attacks

The related-key attack [14] can be either a chosen plaintext attack or low-complexity chosen key attack. In addition, one of the features of this attack is that it is independent of the number of the rounds in the cipher. In many block ciphers, the key scheduling algorithm can be considered as a set of algorithms, and each of the algorithms is used to derive one particular subkey from the subkeys of previous few rounds. If all the algorithms of deriving the subkeys used in the different rounds of the cipher are the same, then all the subkeys generated from a given key can be shifted one round backwards so that a new set of valid subkeys which can be generated from another cipher key can be obtained. These are so called related-keys defined in [14].

The chosen key attacks can choose the relations between the related keys to extract the secret key information themselves. In other words, only the relationships of the keys are known to the attackers while the key information itself is unknown. When it comes to the key schedule of our new cipher, we can see that it is also resistant to the key-related attack, because the algorithms used to derive the subkey for particular rounds are not all the same. In our key schedule, the permutation and four selected bit inversion would be performed in most of the key rounds while the second, fifth, sixth and eighth key round are without the four selected bit inversions. This non-regularity in the key schedule algorithm allows our new cipher to provide resistance against the related-key attack.

4.4. Weak Keys

Weak keys are keys for which all subkeys are identical in the encryption/decryption rounds of a cipher [13]. For semi-weak keys, there is a repetition of different subkeys used in the whole encryption/decryption process of a cipher. Weak keys, if they exist, usually represent a small part of the whole keyspace. In our key schedule of PUFFIN, due to the use of four selected bit inversions and the permutation of the key bits, there are no weak keys existing in our keyspace.

5. Hardware Implementation of PUFFIN

Our new compact block cipher PUFFIN has been designed for very efficient hardware implementations. The 4×4 S-boxes applied in the encryption or decryption process are easy to implement in hardware with simple combinational logic of 4-bit Boolean

functions. All the components are involutional so that the encryption and decryption can share the same hardware. In addition, a very simple key schedule has been applied with only a permutation and bit inversions involved and it costs very limited hardware resources. All the subkeys can be generated on-the-fly so key storage is minimal.

We have implemented the compact architecture of the new cipher targeted to ASIC technology, with a 0.18 μm CMOS standard cell library based on the TSMC process. Synopsys Design Analyzer version X-2005.09 has been used as our synthesis tool.

5.1. Encryption/Decryption Architecture

The loop architecture based on iterating one round of PUFFIN is shown in Figure 2 with the datapath for the encryption/decryption process and the key schedule illustrated. The round function of the encryption is composed of three stages: a substitution layer ("S-box"), a key addition layer ("XOR"), and a permutation layer ("P64"). Only one register is inserted in the round function. The first stage of substitution layer is composed of sixteen 4×4 S-boxes which are used in parallel for substitution. The second stage of key addition layer is implemented as bitwise XORs between input data and the corresponding subkey. The permutation layer "P64" is simply implemented as wirings. Since all the components in PUFFIN are involutional, the decryption process can share exactly the same hardware with the encryption process.

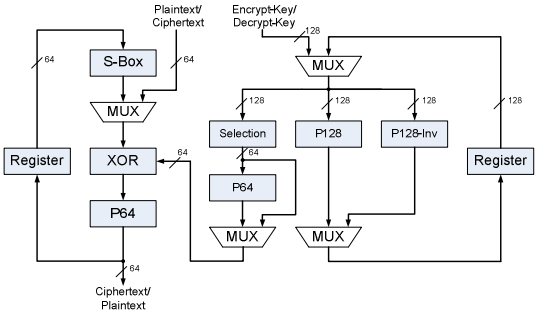

Figure 2. Datapath of PUFFIN

For the architecture of the key scheduling with a key size of 128 bits, each key round function is composed of 128-bit permutation and four selected bit inversions to generate the subkeys used in the encryption process, represented by component "P128". Because the 128-bit permutation used in the key schedule is not involutional, an inverse 128-bit permutation is used in the key round for the decryption process as indicated by component "P128-Inv". The 128-bit permutation layer and inverse 128-bit permutation layer are simply implemented as wirings with the four bit inversions as a few logic gates. The 64 subkey bits selected from the key schedule, can be directly applied in the encryption process, by choosing the right input to the multiplexer below the "Selection" component. However, the involutional permutation operation "P64" designed for the encryption/decryption process should be performed before using the selected subkey bits in the key addition of the decryption process in order to decrypt the ciphertext using the same hardware as encryption.

In order to select 64 bits out of the 128-bit key to be used in the encryption or decryption process, the "Selection" component applies the preferred selection of 64 bits that was discussed in Section 3.3 and presented in Table A3. The "Selection" component requires no logic gates. Since no part of the datapath is shared between the encryption/decryption process and the key schedule, the subkeys used for each round function can be generated on-the-fly.

In order to reduce the number of gates of the PUFFIN datapath, an improved circuit can be implemented, as shown in Figure 3. In this structure, the "Selection" and "P64" components are incorporated into new permutation structures "P128-Comb" and "P128-Inv-Comb". As the permutations only represent wirings, this is easily done with no extra gates, while eliminating the need for the multiplexer at the output of the "Selection" component in Figure 2.

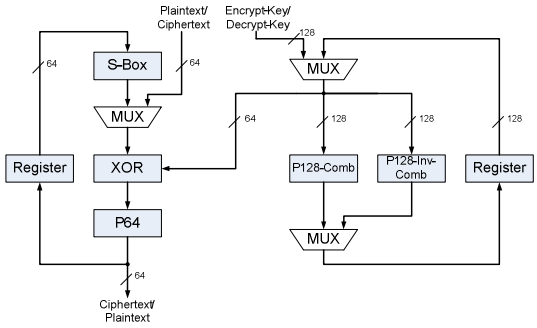

Figure 3. Improved Datapath of PUFFIN

5.2 Hardware Performance Analysis in ASIC

Based on the compact architecture of PUFFIN, we have evaluated the hardware performance in ASIC using a 0.18-μm CMOS standard cell library provided for the TSMC 1P6M process. Table 3 shows the hardware complexity analysis of each component of PUFFIN. All the S-boxes applied in our new cipher are independently implemented and separate logic gates are used to generate each output bit of the S-boxes. We can see that the cost of the S-boxes does not dominate the hardware resources of the datapath, as is typical of AES implementations based on the larger 8×8 S-boxes, only taking about 18% of the hardware area. The major

hardware resources are the registers and multiplexers and they separately take 45% and 30% of the datapath hardware. The remaining XOR components require only about 7% of the hardware.

Table 3. Hardware Complexity Analysis of PUFFIN

Component	Area (gates)	Percentage
S-boxes	475	18%
Register	1152	45%
Multiplexer	769	30%
XOR	181	7%
Datapath	2577	100%

Table 4 shows the comparison of the hardware performance analysis between PUFFIN and several other recently proposed lightweight block ciphers targeted to embedded applications such as smartcards and RFID tags. For comparison in the table we provide the cipher parameters (block size and key size), encryption and/or decryption capabilities, circuit size (in terms of an equivalent number of gates), and throughputs based on the fastest clock and on a reference clock of 100 kHz.

Table 4. Hardware Performance Analysis Comparison

Cipher	Block /Key Size	Enc + Dec	Logic Proc (μm)	Area[4] (gates)	Throughput @ 100kHz (fastest)
PUFFIN	64 /128	Yes	0.18	2577	194 kbps (727 Mbps)
AES-128 [4]	128 /128	Yes	0.35	3400	12.4 kbps (9.9 Mbps)
PRESENT[1] [7]	64 /80	No	0.18	1567 (1991)	200 kbps (658 Mbps[2])
ICEBERG [17]	64 /128	Yes	0.18	5817	400 kbps (552 Mbps)
DESXL[3] [6]	64 /118	?	0.18	2168	44.4 kbps (unknown)
HIGHT [9]	64 /128	No	0.25	3048	188 kbps (151 Mbps)
mCrypton [8]	64 /96	Yes	0.13	3789	492 kbps (unknown)
	64 /96	No	0.13	2681	492 kbps (unknown)

[1] Area results for PRESENT given in [7] total to 1567 gates, while a study undertaken by authors of this paper found the area of PRESENT to be 1991 gates. It appears that this can be explained by the fact that the results from [7] do not seem to include the multiplexers.

[2] Results determined by authors of this paper.

[3] Reference [6] does not clarify whether implementation results are based on encryption-only or are for encryption/decryption architecture. Although DESL requires only 1850 gates, the small key size of 56 bits makes it unsuitable for most applications.

[4] It is not clear whether all areas specified for the different ciphers include circuitry needed for the state machine to control the iterative designs. For PUFFIN and PRESENT-80 in [7], the area specified does not include control circuitry. However, in any case, the required state machine is very small and likely to add no more than about 5% to the area of any cipher.

From the table, several conclusions can be drawn. PUFFIN is a compact design that provides both encryption and decryption in one circuit with a small area. It has a strong security level of 128 bit key size and is capable of high throughput, both in terms of the reference clock of 100 kHz and the fastest possible clock. Although PRESENT-80 takes less area, it does so with the penalty of a smaller key size and without providing decryption capability. Other ciphers that are comparable in security (eg. ICEBERG and mCrypton) require more area to implement. DESXL, while being marginally smaller, is slower by a factor of about 5. Faster ciphers, such as mCrypton and ICEBERG, require more circuit area.

6. Conclusion

In this paper, we have proposed a new compact block cipher PUFFIN based on an involutional SPN structure. This new cipher features involutional operations resulting in an identical datapath for both encryption and decryption and has a simple key schedule capable of on-the-fly subkey generation. The result is a compact cipher design capable of encryption and decryption using a 128 bit key.

For the ASIC implementation based on a 0.18-micron CMOS standard cell design, PUFFIN requires only 2600 gates and can achieve throughputs up to 700 Mbps. Compared to other compact and lightweight block ciphers, it is fair to conclude that a PUFFIN implementation is small, fully capable of supporting modes requiring both encryption and decryption and has a high security level based on a 128 bit key.

Acknowledgements

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

References

[1] A. Menezes, P.C. van Oorschot, and S. Vanstone, *The Handbook of Applied Cryptography*, CRC Press, 1996.

[2] National Institute of Standards and Technology (NIST), "Advanced Encryption Standard (AES)", Federal Information Processing Standard (FIPS) 197, Nov. 2001. Available at csrc.nist.gov/publications.

[3] T. Eisenbarth, S. Kumar, C. Paar, A. Poschmann, and L. Uhsadel, "A Survey of Lightweight Cryptography Implementations", *IEEE Design and Test*, vol. 24, no. 6, Nov. 2007, pp. 522-533.

[4] M. Feldhofer, J. Wolkerstorfer, and V. Rijmen, "AES Implementation on a Grain of Sand", *IEE Proceedings*, vol. 152, no. 1, 2005, pp. 13-20.

[5] F. Standaert, G. Piret, G. Rouvroy, J. Quisquater, and J. Legat, “ICEBERG: an Involutional Cipher Efficient for Block Encryption in Reconfigurable Hardware”, *Fast Software Encryption (FSE 2004)*, LNCS 3017, Springer-Verlag, 2004, pp. 279-299.

[6] G. Leander, C. Paar, A. Poschmann, and K. Schramm, "New Lightweight DES Variants", *Fast Software Encryption (FSE 2007)*, LNCS 4593, Springer-Verlag, 2007, pp. 196-210.

[7] 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)*, LNCS 4727, Springer-Verlag, 2007 pp. 450-466.

[8] C.H. Loon and T. Korkishko, "mCrypton - A Lightweight Block Cipher for Security of Low-Cost RFID Tags and Sensors", *Information Security Applications (WISA 2005)*, LNCS 3786, Springer-Verlag, 2006, pp. 243-258.

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

[10] A. Youssef, S.E. Tavares, and H.M. Heys, "A New Class of Substitution Permutation Networks", *Proceedings of Workshop on Selected Areas in Cryptography (SAC '96)*, Queen's University, Kingston, Ontario, Aug. 1996.

[11] P. Barreto and V. Rijmen, "The Anubis Block Cipher", submitted to the NESSIE Project at www.cosic.esat.kuleuven.be/nessie.

[12] J.B. Kam and G.I. Davida, "Structured Design of Substitution-Permutation Encryption Networks", *IEEE Transactions on Computers*, vol. C-28, no. 10, 1979, pp. 747-753.

[13] 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, 1987, pp. 262-273.

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

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

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

[17] H. Cheng and H.M. Heys, "Compact ASIC Implementation of the ICEBERG Block Cipher with Concurrent Error Detection", *Int'l Symposium on Circuits and Systems (ISCAS 2008)*, Seattle, Wash., May 2008.

Appendix

Table A1. 64-bit permutation ("P64")
(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

Table A2. 128-bit Permutation Used in Key Schedule for Encryption ("P128") (input = row × 8 + column + 1)

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	22	121	126	110	79	81	116	55
1	113	21	29	20	56	76	41	112
2	45	109	95	87	94	44	68	8
3	115	69	6	75	83	5	54	70
4	23	61	106	103	85	124	111	52
5	119	32	100	17	15	34	128	91
6	58	99	120	67	31	98	53	71
7	92	25	38	93	65	2	37	28
8	24	82	88	14	96	118	1	9
9	125	27	127	18	4	10	102	7
10	35	105	48	63	30	77	72	50
11	108	73	12	19	107	11	26	84
12	47	97	117	49	46	33	16	42
13	39	57	114	62	123	101	80	13
14	51	122	64	89	43	60	40	3
15	86	90	59	74	78	104	36	66

Table A3. 64-bit Key Selection ("Selection")
(input = row × 8 + column + 1)

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	3	123	15	58	89	36	98	52
1	57	63	100	70	46	71	94	51
2	83	14	4	22	32	114	84	101
3	12	23	31	65	41	96	120	50
4	45	54	112	122	29	81	30	121
5	97	55	26	64	24	117	19	9
6	111	18	44	86	16	95	42	72
7	2	91	118	124	38	48	43	39