Computers and Electrical Engineering 38 (2012) 148–160

Contents lists available at SciVerse ScienceDirect

Computers and Electrical Engineering

journal homepage: www.elsevier.com/locate/compeleceng

A comparative study of hardware architectures for lightweight block ciphers ☆

Paris Kitsos [a,*], Nicolas Sklavos [b], Maria Parousi [a], Athanassios N. Skodras [a]

[a] Computer Science, Hellenic Open University, Greece
[b] Department of Informatics and M. M., Technological Educational Institute of Patras, Greece

ARTICLE INFO

Article history:
Received 25 August 2010
Received in revised form 24 November 2011
Accepted 24 November 2011
Available online 26 December 2011

ABSTRACT

A hardware-based performance comparison of lightweight block ciphers is conducted in this paper. The DESL, DESXL, CURUPIRA-1, CURUPIRA-2, HIGHT, PUFFIN, PRESENT and XTEA block ciphers have been employed in this comparison. Our objective is to survey what ciphers are suitable for security in Radio Frequency Identification (RFID) and other security applications with demanding area restrictions. A general architecture option has been followed for the implementation of all ciphers. Specifically, a loop architecture has been used, where one basic round is used iteratively. The basic performance metrics are the area, power consumption and hardware resource cost associated with the implementation resulting throughput of each cipher. The most compact cipher is the 80-bit PRESENT block cipher with a count of 1704 GEs and 206.4 Kbps, while the largest in area cipher is the CURUPIRA-1. The CURUPIRA-1 cipher consumes the highest power of 118.1 μW, while the PRESENT cipher consumes the lowest power of 20 μW. All measurements have been taken at a 100 kHz clock frequency.

1. Introduction

Radio Frequency Identification (RFID) readers can read hundreds of tags per second and they do not require the line of sight, as for example a bar code scanner, thus allowing for fast automation of the reading process [1]. Likewise, RFID applications are battery-less, supplying voltage derived from the RF field, resulting in very low power applications compared with the bar code scanners.

RFID tags respond to any reader request within range. Consequently, a person carrying a tagged item effectively broadcasts a fixed identifier to nearby readers. So, anyone with a reader can read the information in the tag, potentially violating the owner's privacy [2].

Moreover, RFID applications have very limited resources, for example, tag memory is restricted to several hundred bits, and approximately 250–5000 logic gates out of the total tag space can be devoted for security-related tasks [3,4]. The block ciphers used for security are lightweight because they aim to reduce the hardware resources needed.

A software implementation requires multiply instructions and more clock cycles to execute the partial operations, increasing the total consumed energy. In addition, the cryptography algorithms demand high processing capabilities. Implementations with a typical microcontroller instruction set, leads to large code size and high power requirements. The main scope is the designing of the appropriate hardware primitives that make these operations more efficient in terms of energy and area resources consumed.

☆ Reviews processed and approved for publication by Editor-in-Chief Dr. Manu Malek.
* Corresponding author. Tel.: +30 2610367535; fax: +30 2610367528.
E-mail addresses: pkitsos@eap.gr, pkitsos@ieee.org (P. Kitsos), nsklavos@ieee.org (N. Sklavos), skodras@eap.gr (A.N. Skodras).

0045-7906/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.compeleceng.2011.11.022

A quite huge range of ciphers were implemented, starting with DESL [5] providing a lower security level with 56-bit key, and continuing with PRESENT [10], which provides medium level of security with 80-bit key and 64-bit block size. Then CURUPIRA family (CURUPIRA-1 [6] and CURUPIRA-2 [7]) provides better security with a 96-bit key and block size. The HIGHT [8], PUFFIN [9] and XTEA [11] ciphers provide a better level of security with 128-bit key and 64-bit block size. Last, DESXL [5] was implemented that uses 184-bit key and 64-bit block size. DESL, DESXL, HIGHT and XTEA are Feistel ciphers [12]. CURUPIRA-1, CURUPIRA-2, PUFFIN and PRESET are Substitution-Permutation network (SP-network) ciphers [13]. However there are some others lightweight block ciphers such us mCrypton [14] and Hummingbird [15] that are not considered in this paper.

A similar work has recently appeared in the open literature [16]. In that work a selection of lightweight cryptography implementations is given. It covers hardware and software implementations of symmetric algorithms like DESL, DESXL and PRESENT block ciphers. There are no comparisons in terms of power consumption (for the hardware implementations) because different standard cell technologies are used. However, the approach of the present paper is entirely different. All the used block ciphers have been implemented in hardware with the same standard cell technology and under the same design philosophy. This confirms fair comparisons and ensures the accuracy of the comparative results.

The organization of the paper is as follows: In Section 2, the design considerations are given. In Section 3, the hardware implementations of the used ciphers are reported. The detailed specifications of the ciphers are reported in [5–11]. The synthesis results, the performance analysis and comparisons are presented in Section 4. Finally, concluding remarks are given in Section 5.

2. Design considerations

The characteristics of the ciphers used in this work are very different and sometimes contradictory. Firstly, the Feistel ciphers modify only half of the block in each round, while the SP-network ciphers modify the complete block in each round. Also, some substitution boxes (S-boxes), for example in CURUPIRA-1 and CURUPIRA-2 ciphers, are implemented in either combinational logic or with memory elements, while other S-boxes (DESL, DESXL, PUFFIN and PRESENT ciphers) may be implemented only using memory elements as their combinational equations are not given by the ciphers' constructors. The major problem with the S-boxes operation is that it cannot be easily encoded in a linear equation. Finally, some ciphers (CURUPIRA-1 and CURUPIRA-2) require matrix multiplications, while others (DESL, DESXL and HIGHT) not. For the above reasons a general architecture option for all ciphers' implementations was used and no algorithm specific hardware designs were used for each cipher. It is impossible to apply the same design optimizations to all ciphers in order to examine the optimization methods efficiently and accurately.

The iterative looping architecture was used for all ciphers. This is because the payload data transferred in RFID applications is too small and the bit rate is also low. So, the full loop unrolling architecture with successive rounds and pipeline stages between each round is not a practical solution. A general diagram of the architecture is presented in Fig. 1.

As shown in this figure, only one round of each cipher is implemented. The output of the *basic round* unit is buffered and used as input to the next round. During initialization the multiplexer chooses the plaintext and then chooses the output of the basic round unit. In this architecture the key scheduler also consists of one basic round. The produced round subkey is used both for the data encryption/decryption and as input to the next key round. The subkeys are generated on-the-fly. All S-boxes used by the ciphers have been implemented by Look-Up-Tables (LUTs) using ROM blocks. Each round (Basic Round and Key Basic Round) needs one clock cycle for its execution, so this architecture requires n clock cycles to perform encryption/decryption for an n-round cipher.

3. Block ciphers hardware implementations

The hardware implementations of the ciphers used in this work are briefly reported below.

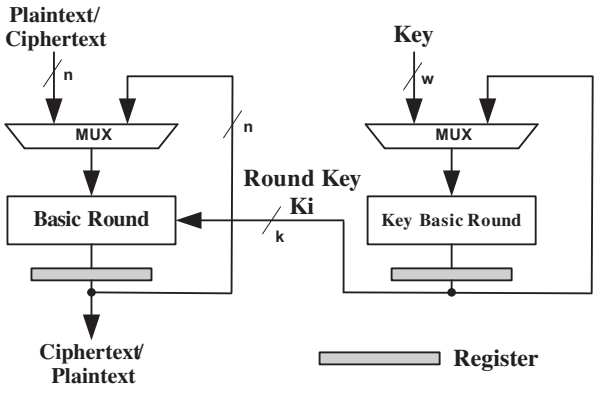

Fig. 1. Block diagram of the used architecture.

3.1. DESL and DESXL hardware implementations

The DESL cipher uses a 64-bit Plaintext/Ciphertext and a 56-bit Key in order to produce a 64-bit Ciphertext/Plaintext. The datapath for the encryption/decryption process and the key scheduling are depicted in Fig. 2.

The basic round for encryption/decryption consists of the function f and a 32-bit XOR gate. The key scheduling mainly consists of two permutations (*PC-1* and *PC-2*) and the Key Basic Round. Two rotator functions (*LS*) compose this round.

The datapath for the encryption/decryption process and the key schedule for DESXL cipher are depicted in Fig. 3. The main difference from DESL is that it uses a 184-bit Key. The 64 most significant bits and the 64 least significant bits are used for key whitening in order to increase the security level of the cipher.

The round subkeys are generated on-the-fly by the key scheduling units for both ciphers. The produced Ciphertext/Plaintext for both ciphers is generated after sixteen rounds.

3.2. CURUPIRA-1 hardware implementation

The CURUPIRA-1 implementation uses 96-bit Key and generates the result after 10 rounds. It consists of the Key Scheduling Unit, which is responsible for the round subkeys generation, and the CURUPIRA-1 Core Unit, which executes the basic encryption/decryption procedure.

The hardware implementation of the CURUPIRA-1 Core Unit is illustrated in Fig. 4. The nonlinear layer γ, is composed of 12 *S-boxes*. Each S-box consists of two 4-bit miniboxes *P* and *Q*. The *P* and *Q* miniboxes are 4-bit to 4-bit boxes that, in hexadecimal notation, are given in the following table:

i	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
P(i)	3	F	E	0	5	4	B	C	D	A	9	6	7	8	2	1
Q(i)	9	E	5	6	A	2	3	C	F	0	4	D	7	B	1	8

The *P* and *Q* mini boxes are implemented by Look-Up-Tables (LUTs). The permutation layer π is implemented by wired shifters. Actually, layer π is a 3×4 matrix with each of its arguments equal to a byte.

The operations that are executed are as follows. The first row of data is kept unchanged. In the second row, the data of the first argument are swapped with the data of the second argument, and the data of the third argument are swapped with the data of the forth argument. Finally in the third row, the data of the first argument are swapped with the data of the third argument and the data of the second argument are swapped with the data of the fourth argument. The diffusion layer θ, is a matrix multiplication between the round state (layer π output) and a generator matrix, *D*. The layer θ input state is a 3×4 matrix. Each argument of this matrix is also one byte. The key addition ($\sigma[k^{(r)}]$) consists of eight 2-input XOR gates for any byte of the state.

The Key Scheduling Unit, which consists of the Key Evolution and the Key Selection, is depicted in Fig. 5.

The Key Evolution Function is mainly comprised of the key addition component, the linear transforms ξ and μ, six 2-to-1 96-bit multiplexers and a 96-bit Register. The key addition consists of eight 2-input XOR gates for any byte of the state. Every

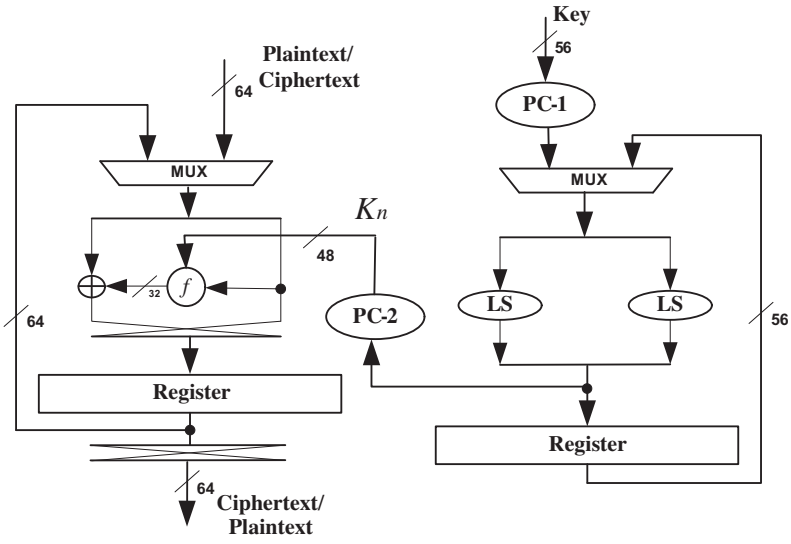

Fig. 2. The DESL hardware implementation.

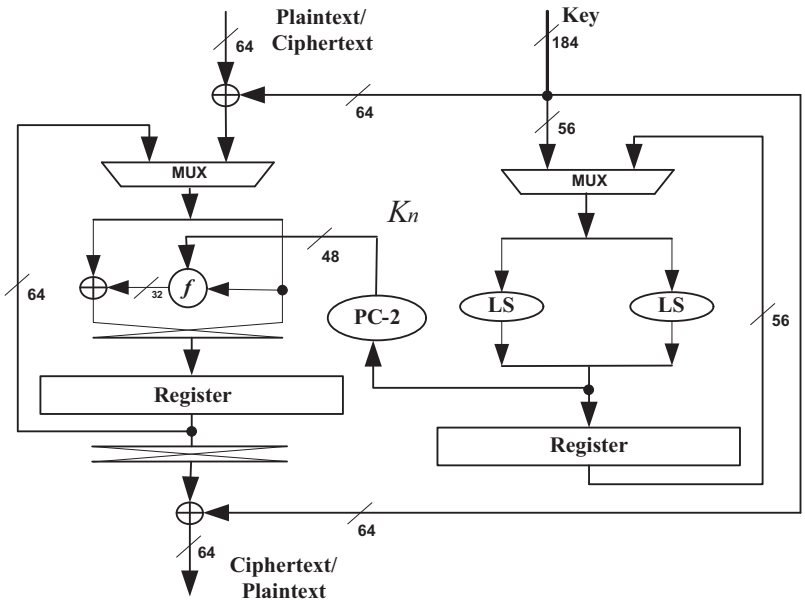

Fig. 3. The DESXL hardware implementation.

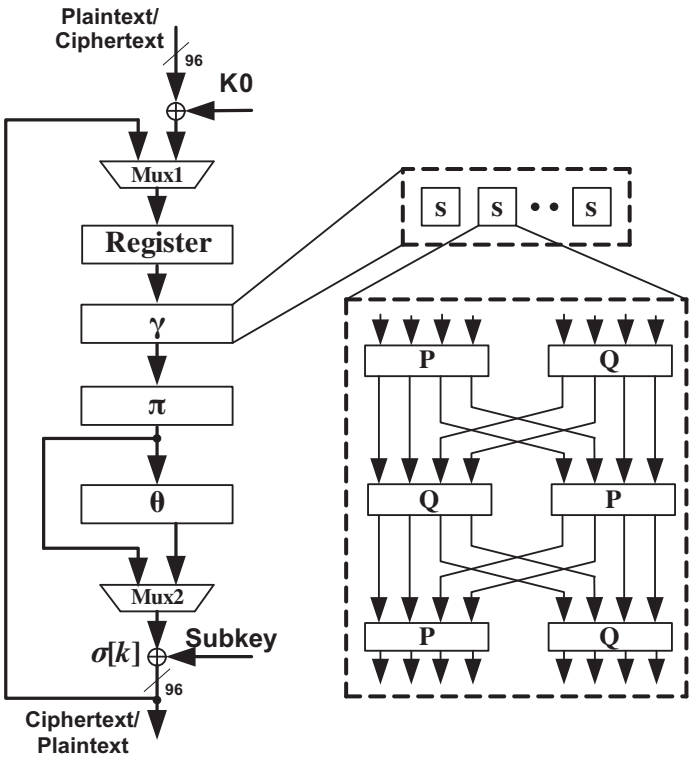

Fig. 4. The CURUPIRA-1 core unit hardware implementation.

bit of the first row of the state is XORed with the appropriate bit of the predefined ($q^{(s)}$) constants. Those constants are realized in hardware as ROM blocks. Then, the linear transform ξ, rotates its input state according to the following rule during the encryption operation: it keeps the first row of its argument unchanged, rotates the second row one position to the left, and rotates the third row one position to the right. During the decryption operation this works reversely. Finally, the transform μ is a matrix multiplication between the input state and a generator matrix E.

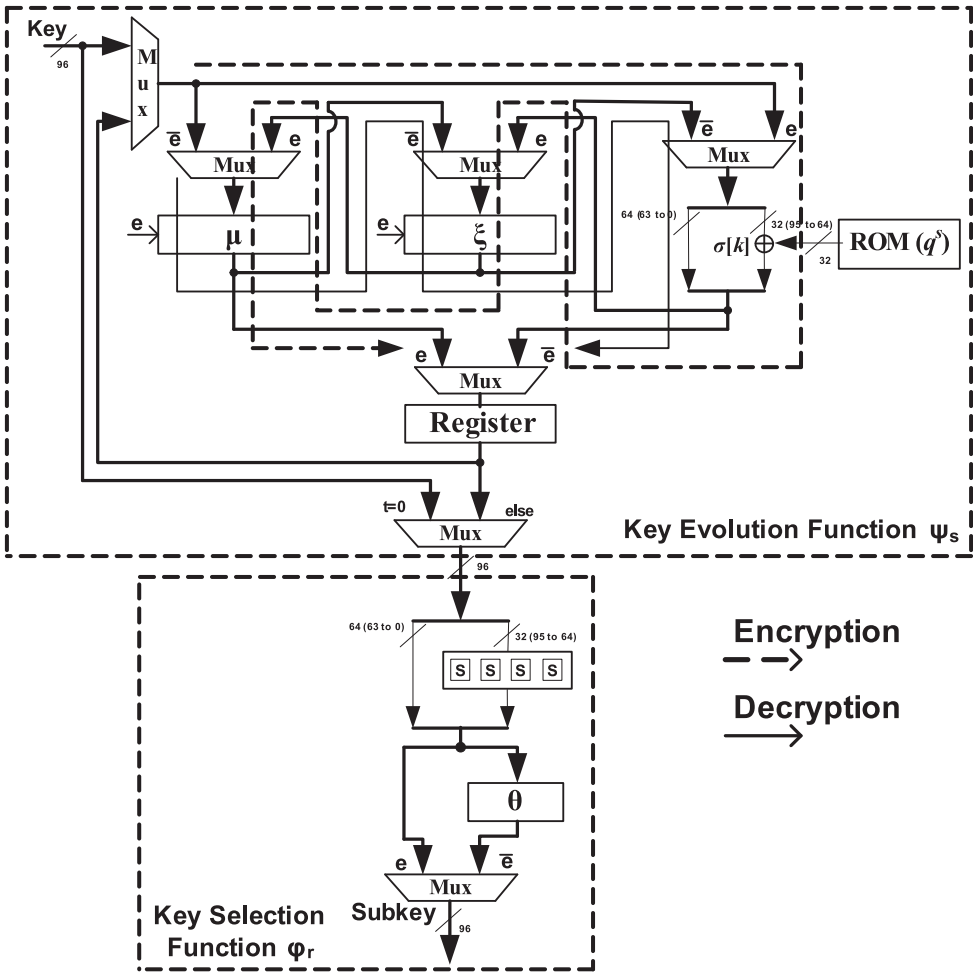

Fig. 5. The CURUPIRA-1 key scheduling unit hardware implementation.

During the encryption the e flag is true ($e = 1$), while during the decryption the e flag is false ($\bar{e} = 0$). The implementation shares two tables, X and $x^2 = X \circ X$, and eight 2-input XOR gates. Symbol "$\circ$" denotes the sequential (associative) operation of each algebraic function where the right function is executed first.The Key Selection Function consists of four S-boxes and a diffusion layer θ. The multiplexer selects the appropriate subkeys for encryption (e) or decryption ($\bar{e}$). The *S-boxes* are applied in the first row (95–64) of the key state.

3.3. CURUPIRA-2 hardware implementation

The CURUPIRA-2 implementation uses also 96-bit Key and generates the result after 10 rounds. It consists of the Key Scheduling Unit, which is responsible for the round keys generation, and the Core Unit, which executes the basic encryption procedure. CURUPIRA-2 block cipher uses exactly the same core unit with CURUPIRA-1 as described in the previous section. The only difference between the two versions of CURUPIRA is in the Key Scheduling Unit.

The Key Scheduling Unit of CURUPIRA-2 is illustrated in Fig. 6.

Similar to CURUPIRA-1, the two basic functions, Key Evolution and Key Selection, are also separate. The Key Evolution Function is mainly comprised of the key addition component, the transform N/N^{-1} [7] (N for encryption and N^{-1} for decryption), six 2-to-1 96-bit multiplexers and a 96-bit register. For the key addition ($\sigma[k^{(r)}]$) a 32-bit XOR gate is used. Then, the transform N/N^{-1} is a matrix multiplication between the input state and the polynomial x^8.

In Fig. 7 the implementation of the N/N^{-1} module is shown.

Its operation is based on byte swapping, XOR operations and the transformations of two auxiliary functions *T1* and *T0* defined by the specifications. The Key Selection Function consists mainly of four *S-boxes* and a diffusion layer θ. During the encryption the e flag is true ($e = 1$), while during the decryption the e flag is false ($\bar{e}$).

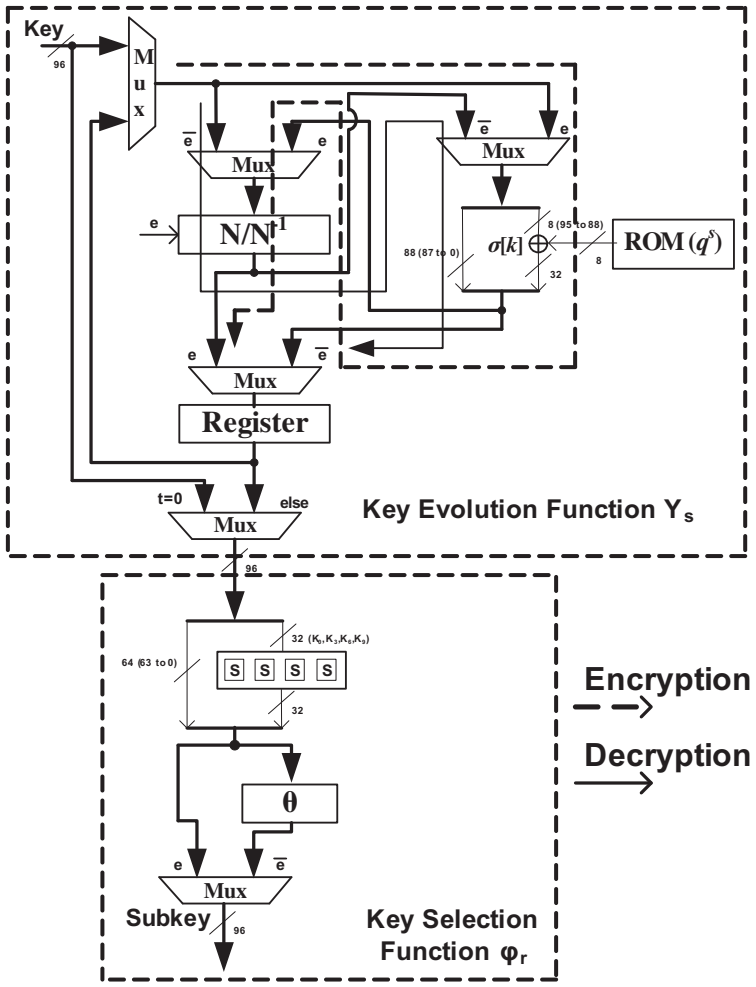

Fig. 6. The CURUPIRA-2 key schedule hardware implementation.

3.4. HIGHT hardware implementation

The encryption/decryption datapath for HIGHT cipher is shown in Fig. 8. It consists of eight parallel branches; each one updates one byte. The construction of each branch begins with a 3×1 8-bit multiplexer (*MUX*). During the initial transformation, each multiplexer selects the second input ('1') as output. Also, during the encryption all the bytes are left shifted and each multiplexer selects the first input ('0') as output and finally, during the decryption all the bytes are right shifted and each multiplexer selects the third input ('2') as output. In the first and fifth branches (from left to right in Fig. 8) there is

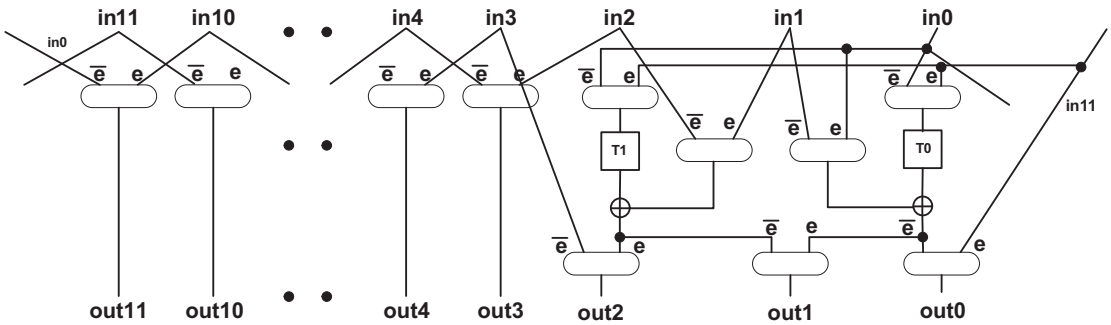

Fig. 7. The N/N-1 module hardware implementation.

an 8-bit XOR gate. In the third and seventh branches there is an adder/subtracter. During the encryption process the adder is selected, while the subtracter is selected during the decryption. All other branches are left without any operation. The 2×1 8-bit multiplexers select either whitening key bytes during the initial and final transformations or the four subkey bytes (SK_i, $i = 0,1,2,3$) used in each round. In the first and third (from left to right) there is an adder and the function F_0, while in the second and fourth there is an XOR gate and the function *F1*. Finally, a 64-bit Register stores temporarily the output of each round.

The hardware architecture of the HIGHT key scheduling unit is shown in Fig. 9a. It consists of two 2×1 128-bit multiplexers. The first one selects either the secret key or the output of the previous computed data. The 128-bit Register stores temporarily the computed data.

The Bytewise Rotation and Inverse Bytewise Rotation are used in order to rotate the 128-bit according to the specification during the encryption and decryption operations, respectively. The Bytewise rotation, rotate each byte by one bit in left direction since the Inverse Bytewise Rotation, rotate each byte by one bit in right direction. Both modules are implemented by simple wirings with no additional cost in hardware resources. The other multiplexer selects the appropriate output to perform encryption or decryption.

For the first four subkeys the 32 least significant bits (31 to 0) are selected. For the next four subkeys bits 63 to 32 are selected, and so on. These 32 bits are added with the δ constants. Finally, the 32-bit subkey is split according to the rule $Subkey = SK_{i+3}SK_{i+2}SK_{i+1}SK_i$ and used as shown in Fig. 8.

The 128 7-bit constants $\delta 0$, $\delta 1$, …, $\delta 127$ are generated by the δ Generation Module, the hardware implementation of which is shown in Fig. 9b. This module mainly consists of two Linear Feedback Shift Registers (*LFSRs*). The left *LFSR*, used for encryption (LFSR h encryption), shifts the data to the right and the right *LFSR*, used for decryption (LFSR h decryption), shifts the data to the left. The main 'trick' of the *LFSRs* implementations is that there are five shift registers working simultaneously. This is because there is a need to take all the necessary *LFSR* states at the same time.

The adders and adder/subtracter are implemented by Ripple-Carry adders in order to reduce the hardware resources.

3.5. PUFFIN hardware implementation

The block diagram of the PUFFIN hardware implementation is given in Fig. 10. The left part is the encryption/decryption datapath, while the right part is the key scheduling unit. The nonlinear layer γ, is composed of 16 S-boxes that are implemented by Look-Up-Tables (LUTs). The S-boxes are 4-bit to 4-bit boxes that, in hexadecimal notation, are given in the following table: .

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

The 2×1 64-bit multiplexer selects either the initial Plaintext/Ciphertext or the output of the previous round. For the key addition a 64-bit XOR gate is used. The permutation layer (actually an 8×8 permutation table), *P64*, is implemented by simple wirings that do not cost any hardware gates. Finally, the 64-bit Register temporally stores the output stage in each round.

The key scheduling unit consists of two parallel datapaths. The first one produces the necessary subkeys during the encryption and the second one produces the subkeys during the decryption. All the subkeys are generated on-the-fly in both the encryption and decryption processes. The *P128-Comb* and *P128-Inv-Comb* permutations (actually 16×8 permutation

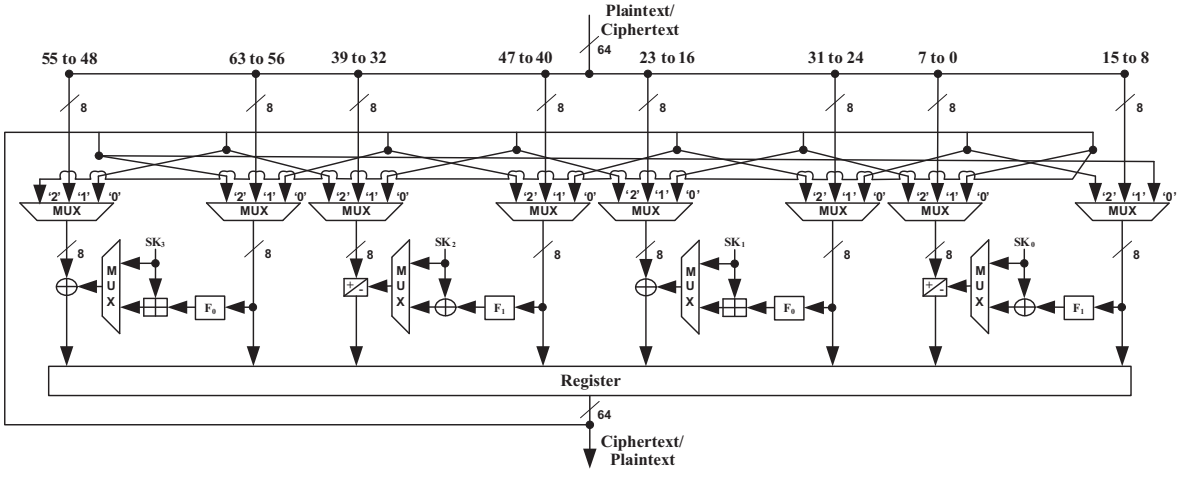

Fig. 8. The HIGHT encryption/decryption datapath hardware implementation.

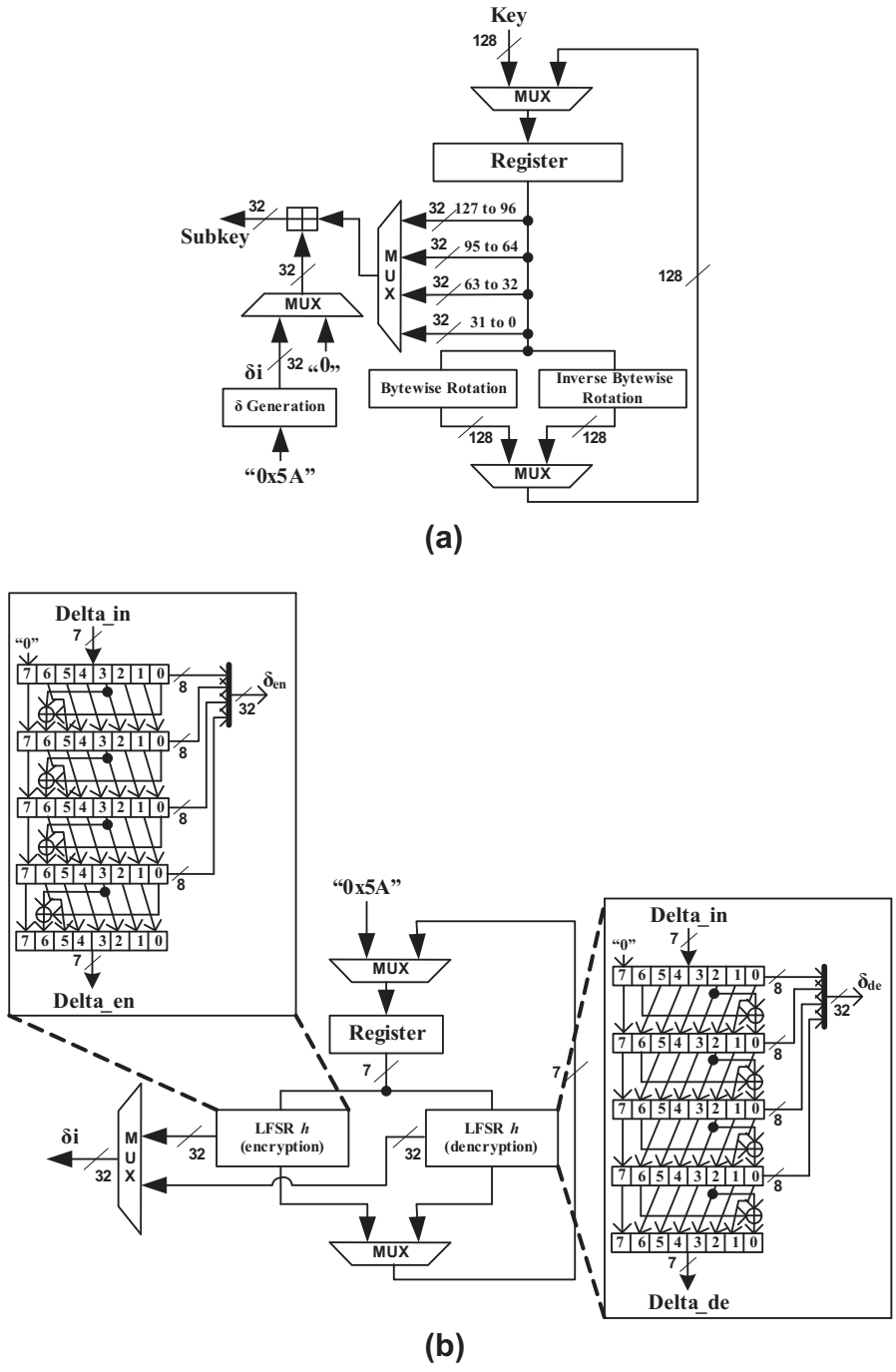

Fig. 9. (a) The HIGHT key schedule hardware implementation, (b) the δ generation module hardware implementation.

tables) are implemented by Look-Up-Tables (LUTs). Finally, the required subkeys are only 64-bit long and, so, an additional key selection is needed in the key schedule to choose 64-bit out of the 128 key bits generated in the key schedule. However, this imposes no cost in hardware.

3.6. PRESENT hardware implementation

Fig. 11 shows the PRESENT hardware architecture where the left part indicates a 64-bit data encryption/decryption block and the right part is a key scheduling for an 80-bit key.

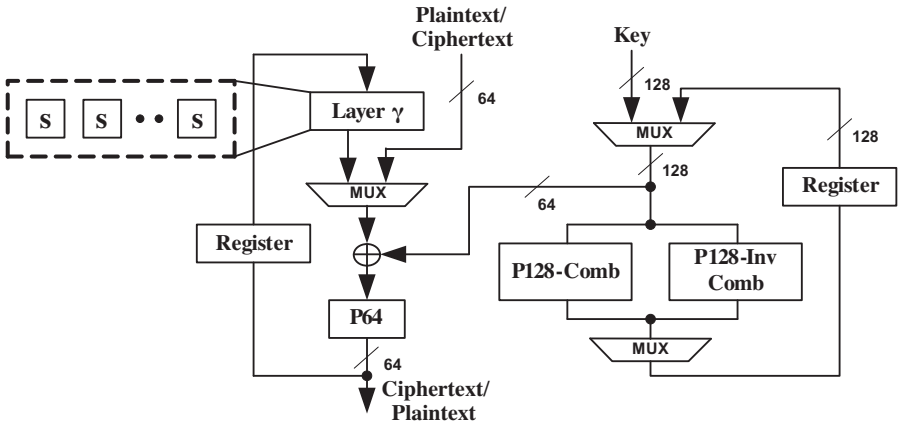

Fig. 10. The PUFFIN hardware implementation.

The *S-Box Layer* is comprised of 16 identical S-boxes (implemented by Look-Up-tables (LUTs)) applied in parallel. The S-box is a 4-bit to 4-bit box that, in hexadecimal notation, is given in the following table:

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

The bit permutations (*p Layer*) are implemented by simple wirings. The bit permutation used in PRESENT is given by the following table with the explanation that bit i is moved to bit position P(i).

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
P(i)	0	16	32	48	1	17	33	49	2	18	34	50	3	19	35	51

i	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
P(i)	4	20	36	52	5	21	37	53	6	22	38	54	7	23	39	55

i	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
P(i)	8	24	40	56	9	25	41	57	10	26	42	58	11	27	43	59

i	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
P(i)	12	28	44	60	13	29	45	61	14	30	46	62	15	31	47	63

The 2×1 64-bit multiplexer selects either the initial Plaintext/Ciphertext or the output of the previous round. Finally, for the key addition a 64-bit XOR gate is used.

The round subkeys are generated on-the-fly. Initially, the value of the 80-bit key Register is left rotated by 61-bit positions. Then, the four most significant bits (79 to 76) are passed through an S-box, and finally the bits $k_{19}k_{18}k_{17}k_{16}k_{15}$ of K are XORed with the least significant bits of *round_counter*.

3.7. XTEA hardware implementation

The hardware implementation of the XTEA cipher is depicted in Fig. 12. In the left part the 64-bit encryption/decryption datapath is presented and in the right part the key scheduling unit for the 128-bit Key.

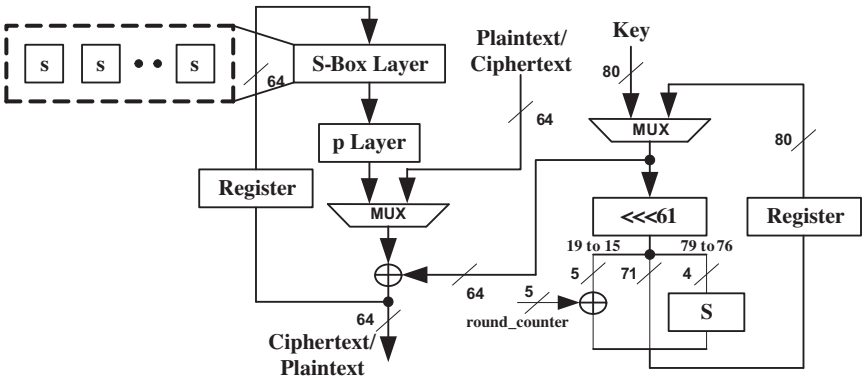

Fig. 11. The PRESENT hardware implementation.

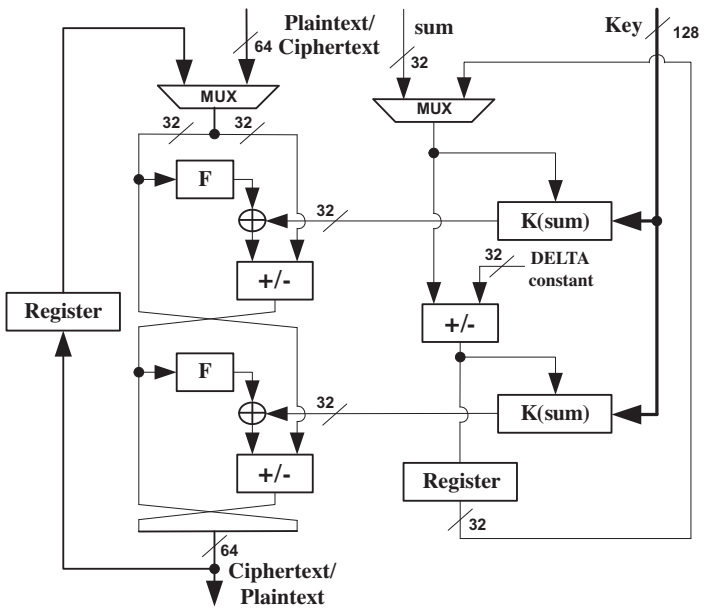

Fig. 12. The XTEA hardware implementation.

The encryption/decryption datapath consists of two subrounds each one using one subkey. The 64-bit input is split into two halves. The 32 most significant bits (denoted as A) and the 32 least significant bits (denoted as B). Each subround comprises the module *F*, a 32-bit adder/subtracter (shown as "+/−" in the figure) and a 32-bit XOR gate. Module *F* implements the permutation function $f(x) = (x << 4 \wedge x >> 5) + x$. Thus it consists of two shifters (shifts the data 4 bits to the left and 5 bits to the right, respectively), a 32-bit XOR gate and a 32-bit adder. The shifters are implemented by simple wiring. During the encryption A is applied to the left branch of the Feistel network and B is applied to the right branch. For decryption the reverse takes place.

In the key scheduling unit, module *K*(*sum*) selects one block out of the four 32-bit blocks that comprise the key, depending on either bits 1 and 0 or bits 12 and 11 of *sum*. The value of *sum* is updated between the first and the second subround. This value is added to *DELTA* during encryption and subtracted from *DELTA* during decryption. The results of module *F* and the *K*(*sum*) are XORed and applied to right branch of the encryption/decryption datapath by addition during encryption or by subtraction during decryption.

All the adders and adder/subtracters used by the cipher modules are implemented as Ripple-Carry adders in order to reduce hardware requirements.

4. Performance analysis and comparisons

All architectures presented in Section 3, were simulated by means of MentorGraphics ModelSim tool in order to verify the correct designs and functionality. For the development (synthesis) Synopsys Design Compiler tool was used. The target tech-

nology library is 90 nm CMOS standard-cell ASIC (with 0.9 V core voltage). This tool is also used to generate the area, timing and power estimation reports. The main effort of the synthesis process was area optimization. The definitions of metrics used are described below:

Area: This metric represents the area normalized to that of one two-input NAND gate. This ratio is expressed in Gate Equivalent (GE). *Cycles per block*: This is the number of clock cycles to compute the plaintext/ciphertext and read out the ciphertext/plaintext. *Throughput*: The rate at which new output is produced with respect to time. The number of ciphertext/plaintext bits is multiplied by the operating frequency and divided by the needed cycles. It is expressed in bits-per-second (bps). *Power*: It consists of two major components: the static power and the dynamic power. The static power is proportional to the area and the fabrication process (due to leakage current) and is denoted as P_{leak}. The dynamic power is proportional to the switching activity and is denoted as P_{dyn}. Both components depend also on the supply voltage. *Throughput-to-area ratio*: Measures the hardware resource cost associated with the implementation resulting throughput.

Table 1 compares the implementation results of the used lightweight block ciphers for 100 kHz clock frequency, which is a typical operating frequency of RFID systems. It is noted that at a 100 kHz clock frequency the static power consumption P_{leak} highly surpasses the dynamic power consumption P_{dyn} by an amount of three orders of magnitude. Thus, the total power consumption is in fact considered equal to P_{leak}. The basic message is that the scaling of operation frequency has a great impact on total power consumption.

The exception is the CURUPIRA-1 for which both P_{leak} and P_{dyn} are of comparable magnitudes and the total power consumption is the sum of the P_{leak} and P_{dyn} (P_{total} = 118.1 μW). The DESL and DESXL ciphers are both variant versions of the original DES cipher. The DESXL occupies up to approximately 10% larger chip area than DESL. This is because DESXL has two additional 64-bit (2 input) XOR gates of the keys at the beginning and the end of encryption and decryption in contrary to the DESL.

While they have the same throughput, the DESL is more efficient in hardware because it has better throughput-to-area ratio. Also, it consumes less power than DESXL.

The only difference between CURUPIRA-1 and CURUPIRA-2 is in the key scheduling unit. CURUPIRA-2 requires 14% fewer hardware resources than CURUPIRA-1. This is because the key scheduling unit of the CURUPIRA-2 consists of simpler operations. Last, CURUPIRA-2 has much lower power dissipation. As a result the CURUPIRA-2 is more efficient in terms of hardware implementation.

The PRESENT cipher has the more compact implementation and the CURUPIRA-1 covers the biggest chip area. This is a result of the ciphers design philosophies. The first one uses only a linear permutation layer and a non-linear substitution layer for the basic round transformation. For the round subkeys a shifting operation, an *S-box* and a 5-bit XOR gate are used. The second one uses much more complicated design. From the area efficiency point of view, PRESENT is followed by PUFFIN, DES variants (DESL and DESXL), XTEA, HIGHT and CURUPIRA-2. This is an obvious result and can be deduced by the hardware implementation figures of the previous section.

CURUPIRA-1 and CURUPIRA-2 outperform all other ciphers in terms of throughput at a clock frequency of 100 kHz. This is explained by the fact that they operate with a bigger data block (96-bit) and fewer clock cycles (10 clock cycles) in order to produce their results. Generally SP-network ciphers, like the CURUPIRA family, require fewer iteration rounds in comparison to Feistel ciphers, like DESL/DESXL, HIGHT and XTEA. After the CURUPIRA family the DES variants follow. Specifically, they need fewer clock cycles (only 16 clock cycles) compared to PRESENT (31 clock cycles), and HIGHT, PUFFIN and XTEA that need 32 clock cycles.

Finally, power dissipation is in direct connection to chip area. So, the ciphers with the smaller chip area (PRESENT and PUFFIN) consume the lowest power (PRESENT being better). CURUPIRA-1 consumes the highest level of power. In terms of power consumption, DESL, DESXL and HIGHT ciphers are also very attractive. The XTEA and CURUPIRA-2 power consumption is much lower than that of CURUPIRA-1.

As shown in Table 1, the dynamic power consumption is extremely lower than static power (except for CURUPIRA-1). The major reasons for this are first the technology (static power drastically increases as the technology improves) and second the

Table 1
Comparison of Lightweight Block Ciphers.

Cipher	Block Size (Bits)	Key Size (Bits)	Cycles per block	Area (GEs*)	Throughput at 100 kHz (Kbps)	Throughput-to-area ratio (Kbps/GEs)	Power	
							Pleak(μW)	Pdyn(nW)
DESL	64	56	16	2762	400	0.1448	30	275.5
DESXL	64	184	16	3082	400	0.1297	36	303
CURUPIRA-1	96	96	10	8334	960	0.1151	117	$1.1\ 10^3$
CURUPIRA-2	96	96	10	7334	960	0.1308	98	750
HIGHT	64	128	32	3901	200	0.0512	44.5	511.5
PUFFIN	64	128	32	2303	200	0.0868	25	318
PRESENT	64	80	31	1704	206.4	0.1211	20	242
XTEA	64	128	32	3490	200	0.0573	61	438.5

* Area is given in terms of equivalent two-input NAND gates.

Table 2
Comparisons with Previous Implementations.

Cipher	Cycles per block	Area (GEs)	Power (μW)	Technology
Proposed DESL	16	2762	30	90 nm
DESL [5]	144	1848	0.89 μA	0.18 μm
Proposed DEXL	16	3082	36	90 nm
DESXL [5]	144	2168	-	0.18 μm
Proposed HIGHT	32	3901	44.5	90 nm
HIGHT [8]	32	3048	-	0.25 μm
Proposed PUFFIN	32	2303	25	90 nm
PUFFIN [9]	32	2577		0.18 μm
Proposed PRESENT	31	1704	20	90 nm
PRESENT [10]	32	1570	5	0.18 μm
PRESENT [19]	32	1705	77.1	0.18 μm
Proposed XTEA	32	3490	61	90 nm
XTEA [20]	32	2521	1.54	0.13 μm

low clock frequency operation. In applications such as RFID, the operation frequency is up to 100 kHz and submicrometer technologies are used, so the leakage power is the dominant component of the total power consumption [17], [18]. At the same time, increasing the degree of parallelism, the power consumption increases. The fact that the used implementations have no parallelism (for example pipeline structures), helps in reducing the overall power consumption.

According to the throughput-to-area ratio, DESL is the best cipher for hardware implementation. However the most critical constrains in RFID applications are the power and the area resources. So, PRESENT with the lowest power consumption and with the most compact implementation is the best choice for hardware implementation in RFID. The second best is PUFFIN, followed by DESL, DESXL, HIGHT and XTEA. The CURUPIRA family has the worst hardware implementation in RFID because it covers bigger chip area (7334 GEs for CURUPIRA-2 and 8334 for CURUPIRA-1) and consumes higher levels of power.

Having in mind that no algorithm specific hardware designs were used for each cipher, some comparisons with previous published implementations of the block ciphers are presented in Table 2. The implemented architectures for DESL and DESXL of [5] used a 0.18 μm technology. They are more compact compared to the proposed implementation but execute their operation for one data block after 144 clock cycles. Also, DESL in [5] consumes 0.89 μA compared to the 30 μW of the proposed one. In [8] a 0.25 μm was used and the implementation for HIGHT reaches up to 3048 GEs. This implementation has a similar profile with the proposed one. The PUFFIN implementation in [9] covers an area of 2577 GEs with a technology of 0.18 μm. The proposed PUFFIN implementation outperforms the implementation in [9] in terms of GEs. For all the previous ciphers no measurements for power consumption are given. In [10] and [19] two similar implementations for the PRESENT cipher are reported.

Both used 0.18 μm as a technology mapping. The first one operates at a 100 kHz clock frequency and consumes 5 μW, while the second one operates at a 10 MHz clock frequency and consumes 77.1 μW. The implementation in [19] is similar to our PRESENT implementation in terms of GEs, but consumes 74% more power. The implementation in [10] is more compact than the proposed PRESENT and consumes less power. Finally, an XTEA implementation is presented in [20]. This covers an area of 2521 GEs and consumes 1.54 μW in 0.13 μm technology. From the comparisons between PRESENT and XTEA it is clear that the leakage power for the 90 nm technology compares well with previous reported technologies.

Generally, from the above comparisons it is shown that there is a large variation in power consumption depending on the core voltage of the library. Note that power figures are highly technology dependent, therefore a fair comparison is only possible if the same technology is used.

5. Conclusion

Extended comparisons in terms of hardware performance of lightweight block ciphers are given in this paper. The proposed ciphers are appropriate for RFID applications. The best choice for an RFID security system is PRESENT with the lowest power consumption and the more compact implementation. The second best is PUFFIN, followed by DESL, DESXL, HIGHT and XTEA. The CURUPIRA family has the worst hardware implementation in RFID because it covers bigger chip area, (7334 GEs for CURUPIRA-2 and 8334 for CURUPIRA-1) and consumes higher levels of power.

Acknowledgement

This work was funded by the Greek State Scholarships Foundation.

References

[1] Finkenzeller K, RFID Handbook: Fundamentals and Applications in Contactless Smart Cards and Identification. 2nd ed. Wiley; 2003. ch. 01.
[2] Ohkubo M, Suzuki K, Kinoshita S. RFID Privacy Issues and Technical Challenges. Communications of the ACM 2005;48(9):66–71.
[3] Juels A. RFID security and privacy: a research survey. IEEE J. on Selected Areas in Communications 2006;24(2):381–94.

[4] Weis SA, Sarma SE, Rivest RL, Engels DW. Security and Privacy Aspects of Low-Cost Radio Frequency Identification Systems. Security in Pervasive Computing 2003. LNCS-Springer 2802; 2004. p. 201–212.
[5] Leander G, Paar C, Poschmann A, Schramm K, New Lightweight DES Variants. Fast Software Encryption 2007. LNCS-Springer 4593; 2007. p. 196–210.
[6] Barreto PSLM, Simplício M. Jr. CURUPIRA, a Block Cipher for Constrained Platforms. 25th Brazilian Symposium on Computer Networks and Distributed Systems-SBRC 2007, Belém (PA), Brazil; 2007. p. 61–74.
[7] Simplício M Jr, Barreto PSLM, Carvalho TCMB, Margi CB, Naslund M. The CURUPIRA-2 Block Cipher for Constrained Platforms: Specification and Benchmarking', 1st International Workshop on Privacy in Location-Based Applications – PiLBA '08, Malaga, Spain; 2008. p. 123–140.
[8] Hong D, Sung J, Hong S, Lim J, Lee S, Koo B.-S, Lee C, Chang D, Lee J, Jeong K, Kim H, Kim J, Chee S. HIGHT: A New Block Cipher Suitable for Low-resource Device, Cryptographic Hardware and Embedded Systems – CHES 2006, LNCS. Springer. 4249; 2006. p. 46–59.
[9] Cheng H, Heys HM, Wang C. PUFFIN: A Novel Compact Block Cipher Targeted to Embedded Digital Systems. Euromicro Conference on Digital System Design (DSD 2008), Parma, Italy; 2008. p. 383–390.
[10] Bogdanov A, Knudsen LR, Leander G, Paar C, Poschmann A, Robshaw MJB, Seurin Y, Vikkelsoe C. PRESENT: An Ultra-Lightweight Block Cipher, Cryptographic Hardware and Embedded Systems – CHES 2007, LNCS. Springer. 4727; 2007. p. 450–466.
[11] Wheeler D, Needham R. TEA extensions. Technical Report. Cambridge University, England, October; 1997.
[12] Feistel H. Cryptography and Computer Privacy. Scientific American. 228: (5) May; 1973. p. 15–23.
[13] Daemen J. Cipher and hash function design strategies based on linear and differential cryptanalysis. PhD Thesis. K. U. Leuven; March 1995.
[14] Lim CH, Korkishko T. mCrypton – A Lightweight Block Cipher for Security of Low-Cost RFID Tags and Sensors. 6th International Workshop on Information Security Applications WISA 2005. Jeju Island: Korea; August 2005. p. 22–24.
[15] Engels D, Fan X, Gong G, Hu H, Smith E. Hummingbird: Ultra-Lightweight Cryptography for Resource-Constrained Devices', The 1st International Workshop on Lightweight Cryptography for Resource-Constrained Devices (WLC'2010), Tenerife, Canary Islands Spain; January 2010.
[16] Eisenbarth T, Kumar S, Paar C, Poschmann A, Uhsadel L. A Survey of Lightweight-Cryptography Implementations. IEEE Design & Test. 2007; 24 (6): 522–533.
[17] Amirtharajah R, Chandrakasan AP. Self-powered Signal Processing Using Vibration-based Power Generation, IEEE Journal of Solid-State Circuits 1998; 33 (5): 687–695.
[18] Mukhopadhyay S, Mahmoodi H, Neau C, Roy K. Leakage in Nanometer Scale CMOS Circuits. International Symposium on VLSI Technology, Systems, and Applications; April 2003 p. 307–312
[19] Rolfes C, Poschmann A, Leander G, Paar C, Ultra-Lightweight Implementations for Smart Devices – Security for 1000 Gate Equivalents, 8th IFIP WG 8.8/11.2 international conference on Smart Card Research and Advanced Applications, LNCS. Springer. 2008; 189: p. 89–103.
[20] Kaps JP. Chai-Tea, Cryptographic Hardware Implementations of xTEA. 9th International Conference on Cryptology in India: Progress in Cryptology. LNCS. Springer; 2008: 5365. p. 363–375.

Paris Kitsos received the B.Sc. degree in Physics in 1999 and a Ph.D. in 2004 from the Department of Electrical and Computer Engineering, both at the University of Patras. Currently is research fellow with Hellenic Open University. His research interests include VLSI design, algorithms and architectures for data security and efficient circuit implementations. Dr. Kitsos has published more than 70 scientific articles and technical reports, as well as is reviewing manuscripts for International Journals and Conferences/Workshops in the areas of his research. He has participated to international journals and conferences organization, as Program/Technical Committee Member and Guest Editor.

Nicolas Sklavos is an Assistant Professor, Informatics & MM Dept, Technological Educational Institute of Patras, Hellas. He holds an award for his PhD thesis from IFIP VLSI SOC 2003. N. Sklavos has participated to number of European and National projects and serves as evaluator of both European Commission Projects (FP7) and General Secretary of Research and Development. He is the Council's Chair of IEEE Greece GOLD Affinity Group. He is the Editor-in-Chief for the Information Security Journal: A Global Perspective, Taylor & Francis Group and Associate Editor for IEEE Latin America Transactions, and Computers & Electrical Engineering Journal, Elsevier. He was the General Chair of ACM MobiMedia 2007. He has authored up to 100 scientific articles.

Maria Parousi received the degree in Computer Science in 2008 from the School of Science & Technology at the Hellenic Open University. The same year he has implemented the diploma thesis entitled Hardware Implementations of Architectures on Private key Cryptographic Algorithms for Wireless Communications with supervisor Dr. Paris Kitsos. Since January 2002 she works as programmer in Greek company that manufacture business software with programming languages Visual Basic(6), Power Builder (8) and Informix.

Athanassios Skodras studied Physics and Computer Engineering. Since 1986 he has been holding teaching and research positions at Patras University and CTI, Greece. As of 2002 he is a Professor in Digital Systems and Head of Computer Science, Hellenic Open University, Greece. During the academic years 1988-89 and 1996-97 he has been visiting the DEEE, Imperial College, London, UK. His research interests include image and video coding and watermarking, educational technologies and real-time hardware implementations. He has published 120 technical papers in journals and conference proceedings, authored or co-authored six books, three book chapters, and filed two international patents.