Block ciphers PRESENT and SEA in comparison

Tobias Kußmaul[1], Johannes Löffler[1], and Alexander Wiesmaier[1,2]

[1] TU Darmstadt
[2] AGT International

Abstract. In this paper we compare two popular block ciphers named PRESENT and Scalable Encryption Algorithm (SEA). We give a short introduction into PRESENT and SEA, their function and the different kind of possible implementations. For comparing these two block ciphers, we focus on security, code size, number of Gate Equivalent (GE), energy consumption, and throughput. We also compare both algorithms with the most popular block cipher Advanced Encryption Algorithm (AES) as well as two stream ciphers Trivium and Grain.

Keywords: Internet of things (IoT); lightweight block ciphers; PRESENT; SEA

1 Introduction

The interconnection of embedded computer devices like RFID within the existing Internet infrastructure well known by *Internet of Things (IoT)* is currently one of the major topics in modern IT. IoT is expected to offer more connectivity for devices, services, and systems. There is a need to secure the communication that cannot be satisfied with state of the art cryptography like AES because these devices are often very small and have limited power and energy resources. Both cipher algorithms (PRESENT/SEA) focus on these constraints and will be compared in more detail within this paper. PRESENT was designed with hardware performance in mind, whereby SEA's design goal was a good software performance. There is a lot of work that compares block ciphers or stream ciphers among each other. In this paper we will mainly focus on the comparison of PRESENT and SEA and then compare them with the stream ciphers Trivium and Grain in the comparision section.

Our comparison focuses on security, code size, number of Gate Equivalent (GE), energy consumption, and throughput. Block ciphers are classified by *iterated block ciphers* transforming fixed-sized blocks of plain-text into fixed-sized blocks of ciphertext. The input text is processed in multiple rounds of the same structure. In general, round keys are derived from a key which influences the result of each round. In Block Ciphers the coherence of plain-text and ciphertext should be as complex as possible this was specified by Claude Shannon in the paper Communication Theory of Secrecy Systems [1] published in 1949. Shannon defined two properties, confusion and diffusion, that should be fulfilled by a cipher. Confusion means that each character of the ciphertext should depend on several parts of the key. Diffusion means that if we change a character of the plaintext, then several characters of the ciphertext should change. [1]

2 Related Work

There is a growing number of low-cost cryptography and a number of papers dealing with their comparison. The Paper from [2] *Compact Implementation and Performance Evaluation of Block Ciphers* in ATtiny Devices [3] tries to build a uniform comparision platform by using the ATMEL ATtiny45. [3] [3] implement 12 block ciphers including AES, DESL, HEIGHT, IDEA, KASUMI, KATAN, KLEIN, mCrypton, NOEKEON, PRESENT, SEA and TEA on that platform and published the source code as open source. This should serve as a better comparision platform for the future. There is also work done that compares not only block ciphers among each other. [3] covers block ciphers and stream ciphers. For the comparison, they focus on key bits, block bits, cycles per block, throughput at 100 kHz and the area in gate eqivalents the algorithm needs for implementation. They split the algorithms they compared into two groups, hardware and software oriented ciphers. [4] focued mainly on stream ciphers in their work *Hardware results for selected stream cipher candidates*[4].

3 PRESENT

PRESENT is a lightweight block cipher introduced in 2007 by Orange Labs, Ruhr University Bochum and the Technical University of Denmark. [5] Present is designed to met the constraints of IoT specified above. [5] focused on security and hardware efficiency when designing the algorithm. The block size is 64-bit and the key size is either 80-bit or 128-bit. The most compact hardware implementation of PRESENT needs 1570 (GE) (Assumed 32-bit XOR = 80 GE, 32-bit arithmetic ADD = 148 GE, 192-bit FF = 1344 GE and SHIFT = 0 GE) [5] and is therefore competitive with today's leading compact stream ciphers, which need 1300-2600 GE according to [4].

3.1 Algorithm Specifications

PRESENT is a classical substitution permutation network (SPN) consisting of 31 rounds. At first 32 round keys are generated. The first 31 rounds consits of an XOR operation to introduce a round key K_i for $1 \leq i \leq 32$, where K_{32} is used for post-whitening. Post-whitening obfuscates the structure of the linear bitwise permutation and the non-linear substitution layer of round 31. Each of the 31 rounds exists of three operations. First the current round key is applied

[3] http://www.atmel.com/devices/attiny45.aspx

to the block being encrypted. Then an S-Box is performed that holds Shannon's property of confusion.[5] Confusion means that each character of the ciphertext should depend on several parts of the key. The last step of each round is a permutation. We will now explain the operations in more detail. Figs. 1 and 2 give an illustrative presentation.

```
generateRoundKeys()
for i=1 to 31 do
    addRoundKey(STATE, K_i)
    sBoxLayer(STATE)
    pLayer(STATE)
end for
addRoundKey(STATE, 32)
```

Fig. 1: A top-level algorithmic pseudocode of PRESENT (derived from [5])

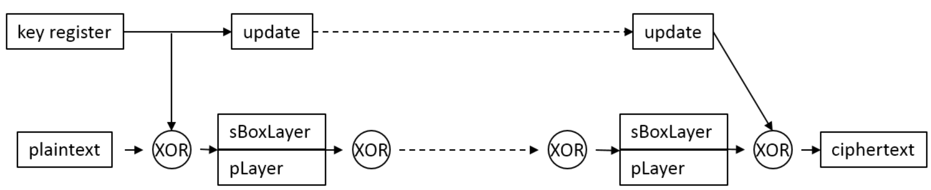


Fig. 2: A top-level algorithmic flowdiagramm of PRESENT (derived from [5])

- *addRoundKey*

  The round key $K_i = K^i_{63} \dots K^i_0$ for $1 \leq i \leq 32$ is applied to the 64-bit block $b_{63} \dots b_0$ for $0 \leq j \leq 63$ by a bitwise exclusive OR.

$$b_j \rightarrow b_j \oplus K^i_j \quad i \in \{1, \dots, 32\},\ j \in \{0, \dots, 63\} \tag{1}$$

- *sBoxLayer*

  PRESENT uses a 4-bit to 4-bit non-linear S-Box. Therefore the current block is considered as a sixteen 4-bit word $w_{15} \dots w_0$ where $w_i = b_{4\cdot i+3}||b_{4\cdot i+2}||b_{4\cdot i+1}||b_{4\dots i}$ for $0 \leq i \leq 15$. The S-Box itself looks as follows

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

– *pLayer*

- In the bit permutation layer the bit i is moved to bit position P(i).

$$P(i) = i \cdot 16 \mod 63 \quad i \in \{0, ..., 63\} \tag{2}$$

– *Key schedule*

PRESENT can be used with either 80- or 128-bit keys. We will focus on the 80-bit key generation. For more information on the 128-bit version, see [5]. The 80-bit key K, is represented as $k_{79}...k_0$. At round i the 64 bit round key $K_i = K_{63}...K_0$ consists of the 64 leftmost bits of the current key K.

$$K_i = K_{63}...K_0 = k_{79}...k_{16} \tag{3}$$

The update routine consists of three parts. First the key register is rotated by 61 bit positions to the left. Then the left most four bits are passed through the S-Box. The last step is an XOR between the least significant bits of the round counter and the bits $k_{19}...k_{15}$ of the key.

$$\begin{aligned} 1.[k_{79}...k_0] &= [k_{18}...k_0 k_{63} k_{19}] \\ 2.[k_{79}k_{78}k_{77}k_{76}] &= S[k_{79}k_{78}k_{77}k_{76}] \\ 3.[k_{19}k_{18}k_{17}k_{16}k_{15}] &= [k_{19}k_{18}k_{17}k_{16}k_{15}] \oplus round_{counter} \end{aligned} \tag{4}$$

The update algorithm for the 120-bit version of PRESENT works almost the same. It takes the leftmost 64-bit into consideration and it has two active S-Boxes in the update routine. For more details see the appendix of the original paper[5].

3.2 Security Analysis

Although it is possible to implement PRESENT both in software and hardware the latter is advised since the major goal for PRESENT when designing it was hardware performance. This aspect is elaborated later. Applications using PRESENT are unlikely to require the encryption of large amounts of data, since the devices it is designed for have low resources.

Two analysis techniques for cryptographic algorithms are differential and linear cryptanalysis. To prove the resistance of PRESENT to those attacks [5] provided a lower bound to the number of so-called active S-boxes in a differential characteristic. This can be captured by the following theorem, proven in [5].

Theorem 1 Any five-round differential characteristic of PRESENT has a minimum of 10 active S-boxes

[5] made four observations to prove the resistance to differential attacks of PRESENT.

- The input bits to an S-Box come from 4 distinct S-Boxes of the same group.
- The input bits to a group of four S-boxes come from 16 different S-Boxes.
- The four output bits from a particular S-Box enter four distinct S-boxes, each of which belongs to a distinct group of S-boxes in the subsequent round.
- The output bits of S-boxes in distinct groups go to distinct S-boxes.

Taking theorem 1 into account, we note that any 25 rounds must have at least 5 x 10 = 50 active S-Boxes. Advanced cryptanalysis techniques allow to remove the outer rounds from a cipher to exploit the characteristic, but the authors think that this is not enough to break up to 25 rounds.

For a *linear attack* it would need about 2^{84} known plaintext/ciphertext pairs to break 31 rounds of PRESENT. Such an amount of data exceeds the available text and is therefore not sufficient at these days. *Structural attacks* are well suited to analyze AES-like ciphers. Such ciphers have word like structures where one word is typically one byte. The bitwise design of PRESENT shall protect against those attacks because the bitwise operations used in the cipher disrupt the word-wise structure.

3.3 Attacks

The first attack on PRESENT is a statistical saturation attack and can be seen as an example of partitioning cryptanalysis. It extracts information about the key by observing non-uniform distributions in the ciphertext [6] and therefore exploits the diffusion properties in block ciphers. It is possible to break up to 15 rounds of PRESENT using $2^{35.6}$ plaintext-ciphertext pairs. The principal attack uses a weakness in the diffusion layer of PRESENT.

Nakahara et al. [7] present a linear algebraic cryptanalysis of reduced round variants for PRESENT. They introduce a pure algebraic cryptanalysis of 5-rounds within that experiment, they were able to recover half of the bits of the key in less than three minutes using an ordinary desktop PC. **??** The attack complexity with respect to time, data (known plaintext), memory, key size for a linear reduced-round attack of PRESENT can be found in Table 4 of [7].

Hernandez-Castro et al. [8] tested the strength of PRESENT's key schedule algorithm of both variants with 80 and 128 bit keys. They used a probabilistic metaheuristic search for semi-equivalent keys, annihilators and entropy minima. Surprisingly, the results show that the 128-bit key seems to be weaker than the 80-bit key. The entropy per byte was 4.006811 (80-bit) compared to 3.744336 (128-bit). The authors affiliated this effect with the theory that there is a reduced number of global optima for the 80-bit version and multiple ones for the 128-bit version [8].

Table 1: List of GE needed for PRESENT implementation (derived from [5])

module	GE	%	module	GE	%
data state	384.39	24.48	KS: key state	480.49	30.61
s-layer	448.45	28.57	KS: S-box	28.03	1.79
p-layer	0	0	KS: Rotation	0	0
counter: state	28.36	1.81	KS: counter-XOR	13.35	0.85
counter: combinatorial	12.35	0.79	key-XOR	170.84	10.88
other	3.67	0.23			
			Overall	**1569.93**	**100**

3.4 Performance

As mentioned before PRESENT requires about 1570 GE when implemented in hardware. In Table 1 a breakdown view of the single components for the hardware implementation can be seen.[5] The most GE are needed to implement the flip-flops for storing the key and the data state, followed by the S-layer and the key XOR. The bit permutation can be implemented using simple wiring and therefore needs no GE. There is a more detailed comparison later in the paper.

4 SEA

Most current block ciphers like AES are designed to only find a good tradeoff between cost, security and performance. SEA on the other hand was designed as a low-cost encryption algorithm running on very limited processing resources. [9] defined the following design goals: low memory, small code size, and limited instruction set. Additionally, they proposed the flexibility as an additional design goal because many block ciphers are designed to run on one specific platform, or processor size and perform very badly is run on a different platform or processor size due to the inflexible design. $SEA_{n,b}$ is parametric in text, key and processor size.

4.1 Algorithm Specifications

One of the stated design goals is that $SEA_{n,b}$ should run on many different platforms, but should behave similar. To achieve this goal, $SEA_{n,b}$ is parametric in the following parameters:

- n: plaintext size, key size
- b: processor size
- $n_b = \frac{n}{2b}$: number of words per Feistel round
- n_r: number of block cipher rounds

The only constraint is that: n has to be a multiple of $6b$. For example for an 8-bit processor there can be the following block ciphers: $SEA_{48,8}$, $SEA_{96,8}$, $SEA_{144,8}$... [9] suggested that the number of rounds is

$$n_r = \frac{3n}{4} + 2 \cdot (n_b + \lfloor \frac{b}{2} \rfloor). \tag{5}$$

This is further explained in section 4.2.

Basic operations $SEA_{n,b}$ only uses a limited number of elementary operations: XOR, S-Box, word rotation, bit rotation and addition $mod 2^b$. They are defined as follows

1. **XOR $\oplus$:**

$$\oplus : \mathbb{Z}_2^{\frac{n}{2}} \times \mathbb{Z}_2^{\frac{n}{2}} \rightarrow \mathbb{Z}_2^{\frac{n}{2}} : x, y \rightarrow z = x \otimes y \Leftrightarrow z(i) = x(i) \oplus y(i) \quad 0 \leq i \leq \frac{n}{2} - 1$$

2. **S-box S:**

$$S : \mathbb{Z}_{2^b}^{n_b} \rightarrow \mathbb{Z}_{2^b}^{n_b} : x \rightarrow x = S(x) \Leftrightarrow \tag{6}$$

$$\begin{aligned} x_{3i} &= (x_{3i+2} \wedge x_{3i+1}) \otimes x_{3i}, \\ x_{3i+1} &= (x_{3i+2} \wedge x_{3i}) \otimes x_{3i+1}, \\ x_{3i+2} &= (x_{3i} \vee x_{3i+1}) \otimes x_{3i+2}, \quad 0 \leq i \leq \frac{n_b}{3} - 1 \end{aligned} \tag{7}$$

where $\wedge$ denotes bitwise AND and $\vee$ denotes bitwise OR.

3. **Word rotation R:**

$$\begin{aligned} R : \mathbb{Z}_{2^b}^{n_b} \rightarrow \mathbb{Z}_{2^b}^{n_b} : x \rightarrow y = R(x) \Leftrightarrow & y_{i+1} = x_i, \quad 0 \leq i \leq n_b - 2, \\ & y_0 = x_{n_b - 1} \end{aligned} \tag{8}$$

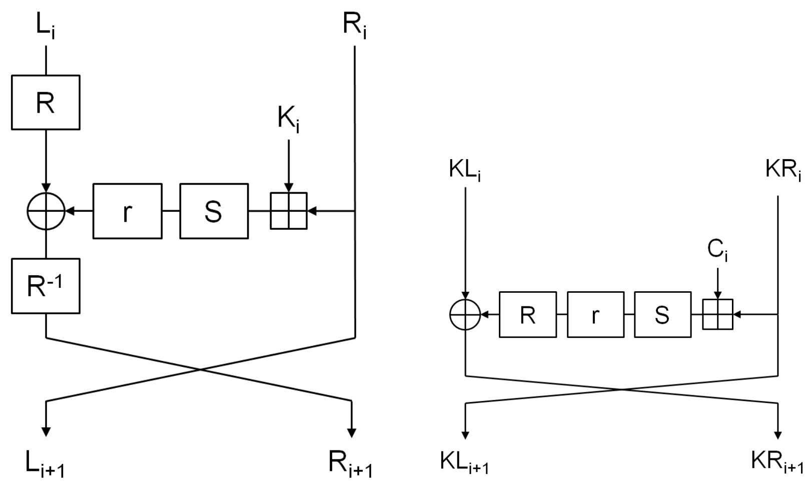
4. **Bit rotation r:**

$$\begin{aligned} r : \mathbb{Z}_{2^b}^{n_b} \rightarrow \mathbb{Z}_{2^b}^{n_b} : x \rightarrow y = r(x) \Leftrightarrow & y_{3i} = x_{3i} \ggg 1, \\ & y_{3i+1} = x_{3i+1}, \\ & y_{3i+2} = x_{3i+1} \lll 1, \quad 0 \leq i \leq \frac{n_b}{3} - 1 \end{aligned} \tag{9}$$

where $\lll$ denotes a left shift and $\ggg$ a right shift.

5. **Addition mod2^b $\boxplus$:**

$$\boxplus : \mathbb{Z}_2^{\frac{n}{2}} \times \mathbb{Z}_2^{\frac{n}{2}} \rightarrow \mathbb{Z}_2^{\frac{n}{2}} : x, y \rightarrow z = x \boxplus y \Leftrightarrow z_i = x_i \boxplus y_i, \quad 0 \leq i \leq n_b - 1 \tag{10}$$



(a) Encryption/Decryption Round (b) Key Round

Fig. 3: SEA Rounds [9]

$SEA_{n,b}$ uses a simple Feistel round for encryption/decryption round as well as for the key round. Figure 3a shows the encryption/decryption round. At the beginning of each round the plaintext is split into two blocks L_i and R_i of n_b words. For encryption the left block L_i is then rotated as describe in equation 8 and xored with the right block R_i as follows using the basic operations introduced earlier:

$$R_{i+1} = R(L_i) \oplus r(S(R_i \boxplus K_i))$$
$$L_{i+1} = R_i$$

During decryption the left block L_i is not rotated as during encryption, instead after the xor the block is rotated using the inverse of the word rotation as describe in equation 8 as follows:

$$R_{i+1} = R^{-1}\left(L_i \oplus r(S(R_i \boxplus K_i))\right)$$
$$L_{i+1} = R_i$$

Figure 3b shows the key round. At the beginning the key is split into blocks KL_i and KR_i of n_b words. The left block KL_i is xored with the right block as follows:

$$KR_{i+1} = KL_i \oplus R(r(S(KR_i \boxplus Ci)))$$
$$KL_{i+1} = KR_i$$

The key schedule is designed so that the key round is the same for encryption as well as decryption. To accomplish this and allow different number of rounds, after $\lfloor \frac{n_r}{2} \rfloor$ rounds the blocks KL_i and KR_i are switch, which leads to reverse the earlier key derivation and lead to the following key expansion:

$$K_0, K_1..., K_{\lfloor \frac{n_r}{2} \rfloor}, K_{\lfloor \frac{n_r}{2} \rfloor - 1}, ...K_1, K_0 \tag{11}$$

4.2 Security Analysis

To ensure resistance against linear and differential cryptanalysis, [9] propose that the number of rounds should be $n_r \geq \frac{3n}{4}$. They further show that to prevent both structural attacks and outer rounds improvements of statistical attacks that SEA is secure if the number of rounds is equal or greater than the number of rounds needed for complete diffusion. For SEA, complete diffusion is achieved after $n_b + \lfloor \frac{b}{2} \rfloor$ rounds. To propagate one active bit to all words it takes at most n_b rounds. This part is done by the combination of the word rotation with the S-box. The diffusion inside each block takes at most $\lfloor \frac{b}{2} \rfloor$ rounds. Getting a more conservative approach [9] propose doubling the number of rounds necessary for complete diffusion. The total number of rounds is

$$\frac{3n}{4} + 2 \cdot (n_b + \lfloor \frac{b}{2} \rfloor). \tag{12}$$

We are not aware of any known attack against SEA.

5 Comparison and Conclusion

SEA and PRESENT share a similar design, but SEA was designed to be implemented in software and PRESENT in hardware. SEA was already implemented in hardware, but implementing PRESENT in software is difficult because of the bitwise permutation that PRESENT uses. In assemble, there is no instruction for this operation which leads to a large performance decrease. Table 2 shows a comparison of PRESENT, SEA and AES. As we can see AES is clearly the fastest one in software, but it also has the highest code size as well as energy consumption.

The speed of each cipher can depend on the amount of GE used in a hardware implementation. As shown in Table 3 PRESENT is faster than AES, but it only uses a 80-bit key where as AES uses a 128-bit one. SEA in comparison to AES is slower as shown in Table 4, but its main advantage is that it is scalable and therefore can be adapted to run on many different platforms.

In Table 3 we compare PRESENT with AES and DES. AES needs about 3400 GE which is well above the amount available in RFID chips. But at the same time, AES offers more security than necessary in a RFID-based application needed. DES in comparison needs about 2309 GE for the hardware implementation. Trivium and Grain are two stream ciphers. Trivium needs about 65% more space than PRESENT whereby Grain is quite small and needs 18% less. 128-bit PRESENT algorithm is at about the same speed as the 80-bit version and needs

Table 2: Comparison of software implementations of ciphers.[2] [3] (At 4MHz).

Cipher	**Block Size**	**Key Size**	**Code Size**	**RAM**	**Cycles**	**Cycles**	**Throughput**	**Energy**
	[bits]	[bits]	[bytes]	[bytes]	[enc+key]	[dec+key]	[Kbps]	[μJ]
AES [2]	128	128	1659	33	4557	7015	-	19.2
AES [3]	128	128	2606	0	6637	7429	77.1	-
PRESENT [2]	64	80	1000	18	11342	13599	-	45.3
PRESENT[3]	64	80	936	0	10723	11239	23.7	-
SEA [2]	96	96	426	24	41604	40860	-	30.3
SEA [3]	96	96	2132	0	7408	9654	39.7	-

about 20 percent more space but therefore is more secure.

At 100 KHz PRESENT achieves a throughput of 200 Kbps while the stream ciphers Trivium and Grain achieve only 100 Kbps, DES 44.4 Kbps, and AES 12.4 Kbps. The ciphers have different block and stream size which are shown in Table 2.

Table 3: Block and stream cipher comparison [3]

Cipher	Key size	Block size	Cycles per block	Throughput at 100 KHz (Kbps)	Logic process	Area GE
			Block Ciphers			
PRESENT-80	80	32	32	200	0.18 μm	1570
AES-128	128	128	1032	12.4	0.35 μm	3400
DES	56	64	144	44.4	0.18 μm	2309
			Stream Ciphers			
Trivium	80	1	1	100	0.13 μm	2599
Grain	80	1	1	100	0.13 μm	1294

SEA was designed to be implemented in software with a reference implementation of 732 bytes [9]. It can also be implemented in hardware. Due to it's similarity to PRESENT it can be assumed that it would take around 2280 GE [10]. Since SEA is parametric the number of gate equivalent for greater plaintext size is higher as shown in Table 4. [11] implemented SEA on a FPGA. The results are shown in Table 4. Comparing $SEA_{144,8}$ with AES-128 we can see that AES is around 14 times faster than $SEA_{144,8}$, but it also requires 12 times the amount of GE and 10 times the power of $SEA_{144,8}$.

The advantages of SEA are its simplicity, its scalability, and the "on-the fly" key derivation. Even though it was originally designed to be implemented

Table 4: Comparison of hardware implementations of ciphers[11].

Cipher	**Plaintext & key size** [bits]	**Word size** [bits]	**Rounds**	**Clock Frequency** [MHz]	**Throughput** [Mbps]	**GE**	**Total Power** [μW]
SEA	96	8	93	250	258	4313	5102.64
SEA	126	7	117	250	269	4565	7216.96
SEA	144	8	135	250	267	6079	8201.22
AES	128	-	10	295	3840	73200	86000.00

in software a number of papers [11] [12] show that it can be implemented in hardware as well with a very good performance.

Our comparison of PRESENT and SEA showed that PRESENT should only be used as hardware implementation while SEA can either be used as software or hardware implementation. Both ciphers have a similar design so they need roughly the same amount of GE, but the performance of SEA increases if more GE are used in the design. In comparison to AES, PRESENT is faster on a small device with very limited resources as shown in Table 3.

References

1. Claude E Shannon. Communication theory of secrecy systems*. *Bell system technical journal*, 28(4):656–715, 1949.
2. Thomas Eisenbarth, Zheng Gong, Tim Güneysu, Stefan Heyse, Sebastiaan Indesteege, Stéphanie Kerckhof, François Koeune, Tomislav Nad, Thomas Plos, Francesco Regazzoni, et al. Compact implementation and performance evaluation of block ciphers in attiny devices. In *Progress in Cryptology-AFRICACRYPT 2012*, pages 172–187. Springer, 2012.
3. Thomas Eisenbarth, Sandeep Kumar, Christof Paar, Axel Poschmann, and Leif Uhsadel. A survey of lightweight-cryptography implementations. *IEEE Design & Test of Computers*, 24(6):522–533, 2007.
4. T Good and M Benaissa. Hardware results for selected stream cipher candidates. *State of the Art of Stream Ciphers*, pages 191–204, 2007.
5. Andrey Bogdanov, Lars R Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew JB Robshaw, Yannick Seurin, and Charlotte Vikkelsoe. *PRESENT: An ultra-lightweight block cipher*. Springer, 2007.
6. Baudoin Collard and F-X Standaert. A statistical saturation attack against the block cipher present. In *Topics in Cryptology–CT-RSA 2009*, pages 195–210. Springer, 2009.
7. Jorge Nakahara Jr, Pouyan Sepehrdad, Bingsheng Zhang, and Meiqin Wang. Linear (hull) and algebraic cryptanalysis of the block cipher present. In *Cryptology and Network Security*, pages 58–75. Springer, 2009.
8. Julio Cesar Hernandez-Castro, Pedro Peris-Lopez, and Jean-Philippe Aumasson. On the key schedule strength of present. In *Data Privacy Management and Autonomous Spontaneous Security*, pages 253–263. Springer, 2012.

9. François-Xavier Standaert, Gilles Piret, Neil Gershenfeld, and Jean-Jacques Quisquater. Sea: A scalable encryption algorithm for small embedded applications. In *Smart Card Research and Advanced Applications*, pages 222–236. Springer, 2006.
10. C Paar, A Poschmann, and MJB Robshaw. New designs in lightweight symmetric encryption. In *RFID Security*, pages 349–371. Springer, 2008.
11. Franois Mace, Franois-Xavier Standaert, Jean-Jacques Quisquater, et al. Asic implementations of the block cipher sea for constrained applications. In *Proceedings of the Third International Conference on RFID Security-RFIDSec*, pages 103–114, 2007.
12. François Macé, F-X Standaert, and J-J Quisquater. Fpga implementation (s) of a scalable encryption algorithm. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 16(2):212–216, 2008.