Block ciphers PRESENT and SEA in comparison

Tobias Kußmaul¹, Johannes Löffler¹, and Alexander Wiesmaier¹,²

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 comparison 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
non 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] 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 equivalents 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 consists 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

\[^3\text{http://www.atmel.com/devices/attiny45.aspx}\]
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.

```plaintext
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])

- **addRoundKey**

  The round key $K_i = K_{i63} \ldots K_{i0}$ for $1 \leq i \leq 32$ is applied to the 64-bit block $b_{63} \ldots b_0$ for $0 \leq j \leq 63$ by a bitwise exclusive OR.

  $$b_j \rightarrow b_j \oplus K_i^j \quad i \in \{1, \ldots, 32\}, \, j \in \{0, \ldots, 63\} \quad (1)$$

- **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 $w_{15} \ldots w_0$ where $w_i = b_{4i+3}||b_{4i+2}||b_{4i+1}||b_{4i}$ for $0 \leq i \leq 15$. The S-Box itself looks as follows

<table>
<thead>
<tr>
<th>x</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>S(x)</td>
<td>C</td>
<td>5</td>
<td>6</td>
<td>B</td>
<td>9</td>
<td>0</td>
<td>A</td>
<td>D</td>
<td>3</td>
<td>E</td>
<td>F</td>
<td>8</td>
<td>4</td>
<td>7</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>
In the bit permutation layer the bit \( i \) is moved to bit position \( P(i) \).

\[
P(i) = i \cdot 16 \mod 63 \quad i \in \{0, \ldots, 63\}
\]

**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 \( k_{79}\ldots k_0 \). At round \( i \) the 64 bit round key \( K_i = K_{63}\ldots K_0 \) consists of the 64 leftmost bits of the current key \( K \).

\[
K_i = K_{63}\ldots K_0 = k_{79}\ldots k_{16}
\]

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 \( k_{19}\ldots k_{15} \) of the key.

1. \([k_{79}\ldots k_0] = [k_{18}\ldots k_0 k_{63} k_{19}]\)
2. \([k_{79} k_{78} k_{77} k_{76}] = S[k_{79} k_{78} k_{77} k_{76}]\)
3. \([k_{19} k_{18} k_{17} k_{16} k_{15}] = [k_{19} k_{18} k_{17} k_{16} k_{15}] \oplus round\_counter\)

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 $5 \times 10 = 50$ 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])

<table>
<thead>
<tr>
<th>module</th>
<th>GE</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>data state</td>
<td>384.39</td>
<td>24.48</td>
</tr>
<tr>
<td>s-layer</td>
<td>448.45</td>
<td>28.57</td>
</tr>
<tr>
<td>p-layer</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>counter: state</td>
<td>28.36</td>
<td>1.81</td>
</tr>
<tr>
<td>counter: combinational</td>
<td>12.35</td>
<td>0.79</td>
</tr>
<tr>
<td>other</td>
<td>3.67</td>
<td>0.23</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>module</th>
<th>GE</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>KS: key state</td>
<td>480.49</td>
<td>30.61</td>
</tr>
<tr>
<td>KS: S-box</td>
<td>28.03</td>
<td>1.79</td>
</tr>
<tr>
<td>KS: Rotation</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>KS: counter-XOR</td>
<td>13.35</td>
<td>0.85</td>
</tr>
<tr>
<td>key-XOR</td>
<td>170.84</td>
<td>10.88</td>
</tr>
<tr>
<td>Overall</td>
<td>1569.93</td>
<td>100</td>
</tr>
</tbody>
</table>

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. 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
- \(n_b = \frac{n}{2^k}\): 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

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

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 $mod2^b$. They are defined as follows

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

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

2. **S-box $S$:**

$$S : \mathbb{Z}_{2^b}^{n_b} \to \mathbb{Z}_{2^b}^{n_b} : x \to y = S(x) \iff \quad (6)$$

$$x_{3i} = (x_{3i+2} \land x_{3i+1}) \oplus x_{3i}, \quad \quad x_{3i+1} = (x_{3i+2} \land x_{3i}) \oplus x_{3i+1}, \quad \quad x_{3i+2} = (x_{3i} \lor x_{3i+1}) \oplus x_{3i+2}, \quad 0 \leq i \leq \frac{n_b}{3} - 1 \quad (7)$$

where $\land$ denotes bitwise AND and $\lor$ denotes bitwise OR.

3. **Word rotation $R$:**

$$R : \mathbb{Z}_{2^b}^{n_b} \to \mathbb{Z}_{2^b}^{n_b} : x \to y = R(x) \iff y_{i+1} = x_i, \quad 0 \leq i \leq n_b - 2, \quad y_0 = x_{n_b-1} \quad (8)$$

4. **Bit rotation $r$:**

$$r : \mathbb{Z}_{2^b}^{n_b} \to \mathbb{Z}_{2^b}^{n_b} : x \to y = r(x) \iff y_{3i} = x_{3i} \gg 1, \quad y_{3i+1} = x_{3i+1}, \quad y_{3i+2} = x_{3i+1} \ll 1, \quad 0 \leq i \leq \frac{n_b}{3} - 1 \quad (9)$$

where $\ll$ denotes a left shift and $\gg$ a right shift.

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

$$\boxplus : \mathbb{Z}_2^n \times \mathbb{Z}_2^n \to \mathbb{Z}_2^n : x, y \to z = x \boxplus y \iff z_i = x_i \boxplus y_i, \quad 0 \leq i \leq n_b - 1 \quad (10)$$
SEA\textsubscript{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:

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

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:

\[
R_{i+1} = R^{-1}(L_i \oplus r(S(R_i \oplus K_i))) \\
L_{i+1} = R_i
\]

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

\[
KR_{i+1} = KL_i \oplus R(r(S(KR_i \oplus C_i))) \\
KL_{i+1} = KR_i
\]

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,
\[ \lfloor \frac{n_r}{2} \rfloor \text{ rounds the blocks } KL_i \text{ and } KR_i \text{ are switch, which leads to reverse the earlier key derivation and lead to the following key expansion:} \]

\[ K_0, K_1, \ldots, K_{\lfloor \frac{n_r}{2} \rfloor}, K_{\lfloor \frac{n_r}{2} \rfloor-1}, \ldots, K_1, K_0 \quad (11) \]

### 4.2 Security Analysis

To ensure resistance against linear and differential cryptanalysis, \[9\] propose that the number of rounds should be \( n_r \geq \frac{3}{4} n_4 \). 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

\[ \frac{3n}{4} + 2 \cdot (n_b + \lfloor \frac{b}{2} \rfloor). \quad (12) \]

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 assembly, 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.\cite{2} \cite{3} (At 4MHz).

<table>
<thead>
<tr>
<th>Cipher</th>
<th>Block Size</th>
<th>Key Size</th>
<th>Code Size</th>
<th>RAM Cycles</th>
<th>Cycles</th>
<th>Throughput</th>
<th>Energy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[bits]</td>
<td>[bits]</td>
<td>[bytes]</td>
<td>[bytes]</td>
<td>[enc+key]</td>
<td>[dec+key]</td>
<td>[Kbps]</td>
</tr>
<tr>
<td>AES [3]</td>
<td>128</td>
<td>128</td>
<td>2606</td>
<td>0</td>
<td>6637</td>
<td>7429</td>
<td>77.1</td>
</tr>
<tr>
<td>PRESENT [2]</td>
<td>64</td>
<td>80</td>
<td>1000</td>
<td>18</td>
<td>11342</td>
<td>13599</td>
<td>-</td>
</tr>
<tr>
<td>PRESENT [3]</td>
<td>64</td>
<td>80</td>
<td>936</td>
<td>0</td>
<td>10723</td>
<td>11239</td>
<td>23.7</td>
</tr>
<tr>
<td>SEA [2]</td>
<td>96</td>
<td>96</td>
<td>426</td>
<td>24</td>
<td>41604</td>
<td>40860</td>
<td>-</td>
</tr>
<tr>
<td>SEA [3]</td>
<td>96</td>
<td>96</td>
<td>2132</td>
<td>0</td>
<td>7408</td>
<td>9654</td>
<td>39.7</td>
</tr>
</tbody>
</table>

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 \cite{3}

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

SEA was designed to be implemented in software with a reference implementation of 732 bytes \cite{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 \cite{10}. Since SEA is parametric the number of gate equivalent for greater plaintext size is higher as shown in Table 4. \cite{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].

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SEA 96 bits</td>
<td>8 bits</td>
<td>93</td>
<td>250</td>
<td>258</td>
<td>4313</td>
<td>5102.64</td>
</tr>
<tr>
<td>SEA 126 bits</td>
<td>7 bits</td>
<td>117</td>
<td>250</td>
<td>269</td>
<td>4565</td>
<td>7216.96</td>
</tr>
<tr>
<td>SEA 144 bits</td>
<td>8 bits</td>
<td>135</td>
<td>250</td>
<td>267</td>
<td>6079</td>
<td>8201.22</td>
</tr>
<tr>
<td>AES 128 bits</td>
<td>-</td>
<td>10</td>
<td>295</td>
<td>3840</td>
<td>73200</td>
<td>86000.00</td>
</tr>
</tbody>
</table>

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


