SEA
a Scalable Encryption Algorithm
for Small Embedded Applications

François-Xavier Standaert[1,3], Gilles Piret[2],
Neil Gershenfeld[3], Jean-Jacques Quisquater[1]

[1]UCL Crypto Group
Laboratoire de Microélectronique, Université Catholique de Louvain
Place du Levant, 3, B-1348 Louvain-La-Neuve, Belgium

[2]Ecole Normale Supérieure, Département d'Informatique,
45, Rue d'Ulm, 75230 Paris cedex 05, France

[3]Center for Bits and Atoms, Massachusetts Institute of Technology
20 Ames Street, Cambridge, MA 02139, USA

{standaert,quisquater}@dice.ucl.ac.be,Gilles.Piret@ens.fr,neilg@cba.mit.edu

Abstract. Most present symmetric encryption algorithms result from a tradeoff between implementation cost and resulting performances. In addition, they generally aim to be implemented efficiently on a large variety of platforms. In this paper, we take an opposite approach and consider a context where we have very limited processing resources and throughput requirements. For this purpose, we propose low-cost encryption routines (*i.e.* with small code size and memory) targeted for processors with a limited instruction set (*i.e.* AND, OR, XOR gates, word rotation and modular addition). The proposed design is parametric in the text, key and processor size, allows efficient combination of encryption/decryption, "on-the-fly" key derivation and its security against a number of recent cryptanalytic techniques is discussed. Target applications for such routines include any context requiring low-cost encryption and/or authentication.

1 Introduction

Resource constrained encryption does not have a long history in symmetric cryptography. Noticeable examples of such ciphers are the Tiny Encryption Algorithm TEA [32] or Yuval's proposal [33]. Both of them are relatively old and their security against attacks such as linear and differential cryptanalysis was hardly evaluated. Present block ciphers, like the Advanced Encryption Standard Rijndael [17, 18] rather focus on finding a good tradeoff between cost, security and performances. While this approach is generally the most convenient, there exist contexts where more specialized ciphers are useful. As a motivating example, ICEBERG [30] is targeted for hardware implementations and shows significant efficiency improvements on these platforms compared to other algorithms.

In this paper, we consequently consider a general context where we have very limited processing resources (*e.g.* a small processor) and throughput requirements. It yields design criteria such as: low memory requirements, small code size, limited instruction set. In addition, we propose the flexibility as another unusual design principle. SEA$_{n,b}$ is parametric in the text, key and processor size. Such an approach was motivated by the fact that many algorithms behave differently on different platforms (*e.g.* 8-bit or 32-bit processors). In opposition, SEA$_{n,b}$ allows to obtain a small encryption routine targeted to any given processor, the security of the cipher being adapted in function of its key size. Beyond these general guidelines, alternative features were wanted, including the efficient combination of encryption and decryption or the ability to derive keys "on the fly". Those goals are particularly relevant in contexts where the same constrained device has to perform encryption and decryption operations (*e.g.* authentication). Finally, the simplicity of SEA$_{n,b}$ makes its implementation straightforward.

Embedded applications such as building infrastructures present a significant opportunity and challenge for such new cryptosystems. For example, introducing programmability into the configuration of lights and switches, thermostats and air handlers, promises to improve the cost of construction, flexibility in occupancy, and energy efficiency of buildings. But meeting this demand on a scale compatible with the economics of the construction industry is going to require secure lightweight implementations of peer-to-peer networks in resource-constrained systems. The Internet-0 approach to end-to-end modulation for interdevice internetworking is typically appropriate in this limit [20]. SEA$_{n,b}$ constitutes a suitable solution for low-cost encryption/authentication within such networks. RFID's or any power/space-limited applications are similarly targeted.

The rest of the paper is structured as follows. Section 2 presents the algorithm specifications. Section 3 discusses security concerns. Performances are evaluated in Section 4 and Conclusions are in Section 5.

2 Specifications

2.1 Parameters and definitions

SEA$_{n,b}$ operates on various text, key and word sizes. It is based on a Feistel structure with a variable number of rounds, and is defined with respect to the following parameters:

- n: plaintext size, key size.
- b: processor (or word) size.
- $n_b = \frac{n}{2b}$: number of words per Feistel branch.
- n_r: number of block cipher rounds.

As only constraint, it is required that n is a multiple of $6b$. For example, using an 8-bit processor, we can derive 48, 96, 144, $\ldots$-bit block ciphers, respectively denoted as SEA$_{48,8}$, SEA$_{96,8}$, SEA$_{144,8}$, ...

Let x be a $\frac{n}{2}$-bit vector. In the following, we will consider two representations:

- Bit representation: $x_b = x(\frac{n}{2}-1)\ x(\frac{n}{2}-2)\ \ldots,\ x(2)\ x(1)\ x(0)$.
- Word representation: $x_W = x_{n_b-1}\ x_{n_b-2}\ \ldots\ x_2\ x_1\ x_0$.

2.2 Basic operations

Due to its simplicity constraints, SEA$_{n,b}$ is based on a limited number of elementary operations (selected for their availability in any processing device) denoted as follows: (1) bitwise XOR $\oplus$, (2) substitution box S, (3) word (left) rotation R and inverse word rotation R^{-1}, (4) bit rotation r, (5) addition $\bmod 2^b$ $\boxplus$. These operations are formally defined as follows:

1. Bitwise XOR $\oplus$: The bitwise XOR is defined on $\frac{n}{2}$-bit vectors:

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

2. Substitution box S: SEA$_{n,b}$ uses the following 3-bit substitution table:

$$S_T := \{0, 5, 6, 7, 4, 3, 1, 2\},$$

in C-like notation. For efficiency purposes, it is applied bitwise to any set of three words of data using the following recursive definition:

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

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

where $\wedge$ and $\vee$ respectively represent the bitwise AND and OR.

3. Word rotation R: The word rotation is defined on n_b-word vectors:

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

4. Bit rotation r: The bit rotation is defined on n_b-word vectors:

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

where $\ggg$ and $\lll$ represent the cyclic right and left shifts inside a word.

5. Addition mod2^b $\boxplus$: The mod 2^b addition is defined on n_b-word vectors:

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

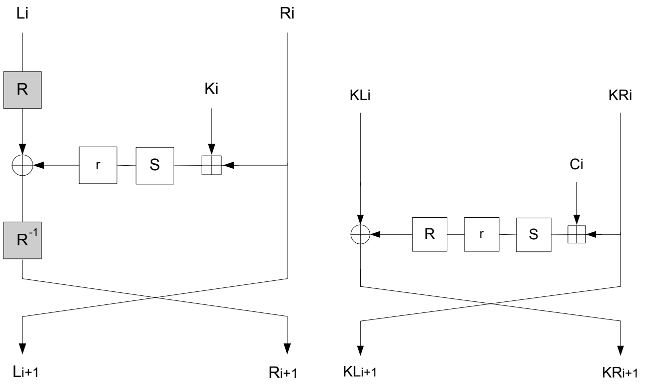

Fig. 1. Encrypt/decrypt round and key round.

2.3 The round and key round

Based on the previous definitions, the encrypt round F_E, decrypt round F_D and key round F_K are pictured in Figure 1 and defined as the functions $F : \mathbb{Z}^2_{2^{n/2}} \times \mathbb{Z}_{2^{n/2}} \rightarrow \mathbb{Z}^2_{2^{n/2}}$ such that:

$$
\begin{aligned}
&[L_{i+1}, R_{i+1}] = F_E(L_i, R_i, K_i) && \Leftrightarrow && R_{i+1} = R(L_i) \oplus r\big(S(R_i \boxplus K_i)\big) \\
& && && L_{i+1} = R_i \\
&[L_{i+1}, R_{i+1}] = F_D(L_i, R_i, K_i) && \Leftrightarrow && R_{i+1} = R^{-1}\Big(L_i \oplus r\big(S(R_i \boxplus K_i)\big)\Big) \\
& && && L_{i+1} = R_i \\
&[KL_{i+1}, KR_{i+1}] = F_K(KL_i, KR_i, C_i) && \Leftrightarrow && KR_{i+1} = KL_i \oplus R\Big(r\big(S(KR_i \boxplus C_i)\big)\Big) \\
& && && KL_{i+1} = KR_i
\end{aligned}
$$

2.4 The complete cipher

The cipher iterates an odd number n_r of rounds. The following pseudo-C code encrypts a plaintext P under a key K and produces a ciphertext C. P, C and K have a parametric bit size n. The operations within the cipher are performed considering parametric b-bit words.

```
$C$=$\mathbf{SEA}_{n,b}(P, K)$
{
    % initialization:
        $L_0 \& R_0 = P$;
        $KL_0 \& KR_0 = K$;
    % key scheduling:
        for $i$ in $1$ to $\lfloor \frac{n_r}{2} \rfloor$
            $[KL_i, KR_i] = F_K(KL_{i-1}, KR_{i-1}, C(i))$;
        switch $KL_{\lfloor \frac{n_r}{2} \rfloor}$, $KR_{\lfloor \frac{n_r}{2} \rfloor}$;
        for $i$ in $\lceil \frac{n_r}{2} \rceil$ to $n_r - 1$
            $[KL_i, KR_i] = F_K(KL_{i-1}, KR_{i-1}, C(r - i))$;
    % encryption:
        for $i$ in 1 to $\lceil \frac{n_r}{2} \rceil$
            $[L_i, R_i] = F_E(L_{i-1}, R_{i-1}, KR_{i-1})$;
        for $i$ in $\lceil \frac{n_r}{2} \rceil + 1$ to $n_r$
            $[L_i, R_i] = F_E(L_{i-1}, R_{i-1}, KL_{i-1})$;
    % final:
        $C = R_{n_r} \& L_{n_r}$;
        switch $KL_{n_r-1}$, $KR_{n_r-1}$;
},
```

where & is the concatenation operator, $KR_{\lfloor \frac{n_r}{2} \rfloor}$ is taken before the switch and $C(i)$ is a n_b-word vector of which all the words have value 0 excepted the LSW that equals i. Decryption is exactly the same, using the decrypt round F_D.

3 Security analysis

3.1 Design properties of the components

Substitution box *S*: The substitution box was searched exhaustively in order to meet the following security and efficiency criteria:

- λ-parameter[1]: 1/2.
- δ-parameter[2]: 1/4.
- Maximum nonlinear order, namely 2.
- Recursive definition.
- Minimum number of instructions.

Remark that, if 3-operand instructions are available, the recursive definition allows to perform the substitution box in 2 operations per word of data. As a comparison, the 3×3 bitwise substitution box used in 3-WAY [15] requires 3. The counterpart of this efficiency is the presence of two fixed points in the table.

Bit and word rotations *r* and *R*: The cyclic rotations were defined in order to provide predictable low-cost diffusion within the cipher, when combined with the bitslice substitution box. It is illustrated in Figure 2 for a single substitution box scheme with parameters $n = 48$, $b = 8$, $n_b = 3$.

Looking at the figure, it can be seen that SEA$_{n,b}$ divides its data in $\frac{2n_b}{3}$ blocks of 3 words. The substitution box is applied in parallel to these blocks. Therefore, the diffusion process (starting with one single active bit in the left branch) is divided into two steps[3]:

[1] We define the *bias* of a linear approximation that holds with probability p as $\epsilon = |p - 1/2|$. The λ*-parameter* of a substitution box is equal to 2 times the bias of its best linear approximation.

[2] The δ-parameter equals the probability of the best differential approximation.

[3] For simplicity purposes, we don't consider the additional diffusion provided by the carry propagation in the $\bmod 2^b$ key addition in this discussion.

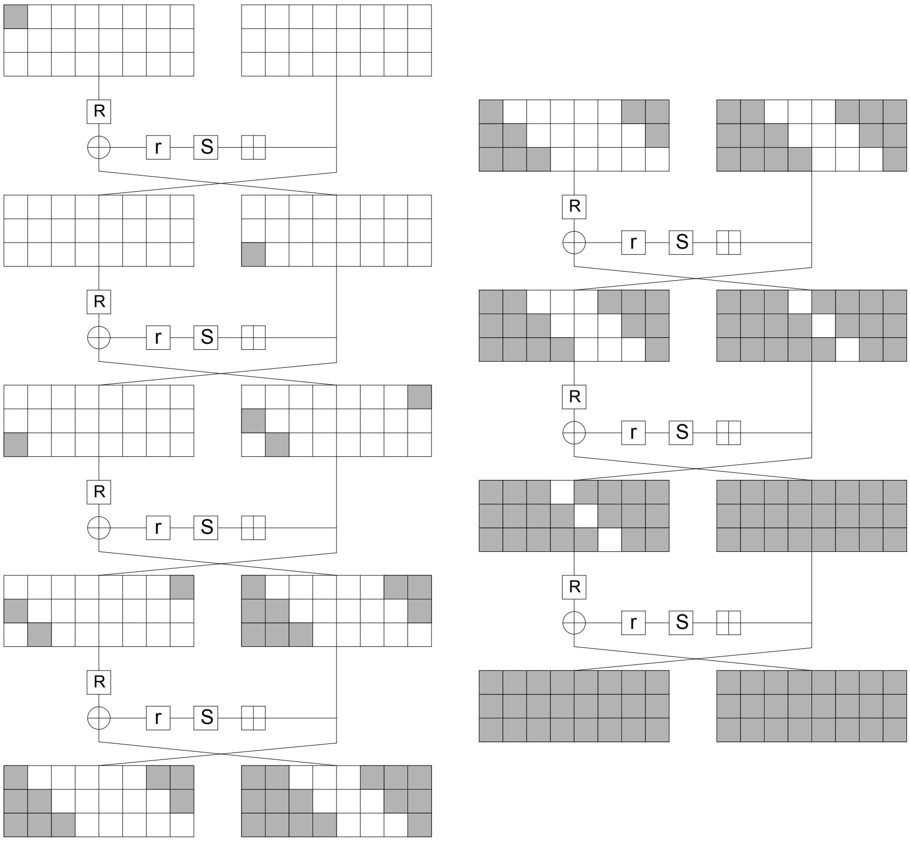

Fig. 2. Diffusion process: grey boxes represent active bits.

- During an initialization step, the single active bit has to be propagated to all the words of the cipher (*e.g.* to our six words in Figure 2).
- During the second step, the diffusion has to be completed within each block.

The first phase is obtained by the combination of the word rotation R (which is the only transform to provide inter-word diffusion) with the substitution box. It requires at most n_b rounds to be completed (in our example, $n_b = 3$ which yields 3 rounds). Once every word has at least one active bit, the combination of r and S yields six more active bits per block in each round. Therefore, finishing the diffusion of all the blocks requires at most $\lfloor b/2 \rfloor$ rounds. Combining these observations, the diffusion is complete after $n_b + \lfloor b/2 \rfloor$ rounds.

Addition mod 2^b $\boxplus$: Using a mod 2^b key addition in place of a bitwise XOR was motivated by different reasons: (1) improvement of the diffusion process, (2) improvement of the non-linearity, (3) same cost/speed as the bitwise XOR in most processors, (4) necessity to avoid structural attacks (see next section).

Overall structure: The overall structure of the cipher follows the Feistel strategy. However, a few points are specific to $\mathrm{SEA}_{n,b}$, namely the key schedule and the position of R, R^{-1} in the encrypt/decrypt rounds.

The key schedule is designed such that the master key is encrypted during half the rounds and decrypted during the other half. It allows to obtain a particular structure of the sequence of round keys such that the key expansion is exactly the same in encryption and decryption. Namely, we have:

$$K_0, K_1, K_2, \ldots, K_{\lfloor \frac{r}{2} \rfloor}, K_{\lfloor \frac{r}{2} \rfloor - 1}, \ldots, K_2, K_1, K_0$$

As a consequence of this structure, the encryption/decryption rounds cannot keep the traditional Feistel structure: it would result in having identical encryption and decryption functions. This is the reason of moving the word rotation to the left branch of the Feistel round.

3.2 Resistance against known attacks

Linear and differential cryptanalysis. From the properties of the substitution box, we can compute bounds for the best linear and differential characteristics through the cipher. We first use the following lemma [29]:

Lemma 1. Let f be the bijective nonlinear function of a 3-round Feistel cipher. Assuming that the linear parameter of f is smaller than λ and its differential parameter is smaller than δ, then the linear, differential parameters of the 3-round cipher Δ, Λ are respectively smaller than λ^2, δ^2.

Since our nonlinear function S has parameter $\delta = 2^{-2}$ and parameter $\lambda = 2^{-1}$, it implies that 3 rounds of $\mathrm{SEA}_{n,b}$ have their linear and differential parameters respectively bounded by $\Delta < 2^{-4}$ and $\Lambda < 2^{-2}$.

However, for a n-bit block cipher, it is respectively required that $\Delta \simeq 2^{-n}$ and $\Lambda \simeq 2^{-\frac{n}{2}}$ to resist against differential [4] and linear cryptanalysis [28]. In order to approach these bounds, we require that:

$$\delta^{2n_r/3} = \left(2^{-2}\right)^{2n_r/3} < 2^{-n} \text{ and } \lambda^{2n_r/3} = \left(2^{-1}\right)^{2n_r/3} < 2^{-\frac{n}{2}}. \qquad (1)$$

In both cases, the required number of rounds is: $n_r \geq 3n/4$.

We note that we used a hybrid approach, between the provable security against linear and differential attacks that consists in bounding the parameter of the best differential/hull, like in lemma 1, and the usual heuristics to estimate the best linear/differential *characteristic* through a cipher (as in the previous estimation for n_r). In fact, the strategy of Equation (1) is similar to the one of *e.g.* the AES Rijndael [17], but we only assume one active s-box per round.

Extensions of linear and differential cryptanalysis. Classical extensions of linear and differential cryptanalysis are non-linear approximations of outer rounds [26], bi-linear cryptanalysis [14], differential-linear cryptanalysis [27], multiple linear cryptanalysis [22, 10], boomerang [31] and rectangle [8] attacks,... However these extensions usually imply only a small improvement compared to the basic attacks. As a matter of fact, non-linear approximations of outer rounds allow to improve the bias of one or two rounds only. Regarding bi-linear cryptanalysis, we quote the author of [14]: *For ciphers similar to DES, based on small substitution boxes, we claim that bi-linear cryptanalysis is very closely related to LC, and we do not expect to find a bi-linear attack much faster than by LC.* It is difficult to evaluate the efficiency of multiple linear cryptanalysis, but it seems more promising for big substitution boxes (as mentioned in [22]). Moreover the improvement on classical cryptanalysis obtained in [10] for the case of DES (which shares with SEA$_{n,b}$ a Feistel structure and a poor diffusion) is limited. Finally, the complexity of differential-linear cryptanalysis and of the boomerang attack and its variants is inherently greater than the one of the basic attacks. As an example, the boomerang (or rectangle) attack allows us to use two short differentials instead of a long one, but using a long differential with probability pq is in general highly preferable to applying a boomerang attack with two short differentials of probability p and q. Therefore although these attacks can perform slightly better in specific cases, the expected improvement is never outstanding. The conclusion is that these extensions actually deserve to be considered in the estimation of the number of rounds necessary to achieve security, but that a reasonable multiplicative factor should be enough to take them into account.

A dedicated related-key attack against a modified version. For $x \in \mathbb{Z}_{2^b}^{n_b}$, we denote by $x \lll a$ the left rotation by a bits of each of the n_b words of x. The non-linear and diffusion layers have the following properties:

- $S(x \lll a) = S(x) \lll a$
- $r(x \lll a) = r(x) \lll a$
- $R(x \lll a) = R(x) \lll a$

Consider a modified version of our cipher where key addition is performed using $\oplus$ rather than modular addition, and where all round constants C_i are such that $C_i \lll a = C_i$, *e.g.* all C_i's equal 0. As a consequence of the previous observations, the modified round F'_E and the key round F_K satisfy:

$$F'_E(L \lll a, R \lll a, K \lll a) = F'_E(L, R, K) \lll a$$
$$F_K(KL \lll a, KR \lll a, 0) = F_K(KL, KR, 0) \lll a$$

These properties are iterative, in the sense that they also hold for the composition of several block cipher rounds. It is immediate to deduce from them a distinguisher on the modified cipher, which requires 2 chosen encryption queries under 2 related keys K and $K \lll a$.

In the actual $\mathrm{SEA}_{n,b}$, the key addition is performed word-wise $\bmod 2^b$. As the property $(X \lll a) \boxplus (K \lll a) = (X \boxplus K) \lll a$ is prevented by certain carry propagations, it only holds with a probability p, which depends on a and the word size b. For $a = 1$, p rapidly converges to $3/8$ as b grows. It is smaller for $1 < a < b-1$. Of course, this probability is averaged for all possible (X, K) and certain keys (*e.g.* "*all zeroes*") yield no carry propagation at all. However, the design properties of the key schedule prevent $\mathrm{SEA}_{n,b}$ from having such weak keys.

Moreover the round constants C_i are generally not such that $C_i \lll a = C_i$ (because they are generated from a counter). Combined with the diffusion in the key schedule, it implies that the similarity between the round keys derived from K and those derived from $K \lll a$ rapidly vanishes.

These properties avoid this structural distinguisher to be propagated through more than a few rounds of $\mathrm{SEA}_{n,b}$.

Square attacks. We explored square attacks [16] on $\mathrm{SEA}_{48,8}$. More precisely, we considered all possible sets of inputs to one branch of the Feistel structure, where the input to some of the substitution boxes is active (*i.e.* takes all possible input values the same number of times), and the input to the other substitution boxes is constant. The other branch is also constant. Therefore the number of plaintexts considered goes from 2^3 (when the input to only one substitution box is active) to 2^{21} (when the input to 7 substitution boxes is active). Our experiments showed that square attacks do not allow to pass through more rounds than the diffusion pattern illustrated in Figure 2. It is expected that it remains the same when different parameters n and b are considered, which implies that $n_b + \lfloor b/2 \rfloor$ rounds are enough to prevent square attacks. Note that although our observations also hold for $\oplus$-$\mathrm{SEA}_{n,b}$, the use of addition mod 2^b provides better resistance against square attacks.

Truncated and impossible differentials. As for square attacks, the diffusion analysis illustrated in Figure 2 provides an estimation of the number of rounds required to prevent truncated differential attacks [25]. Impossible differentials [7] are usually built by concatenating two incompatible truncated differentials. As a consequence, we estimate the number of rounds necessary to prevent the construction of an impossible differential distinguisher as $2 \cdot (n_b + \lfloor b/2 \rfloor)$.

Interpolation attacks. The interpolation attack [21] is possible when the whole cipher can be written as a relatively simple algebraic expression. It requires the substitution box to have a compact expression, and the diffusion layer to permit the composition of these expressions. In the case of $\mathrm{SEA}_{n,b}$, there is a priori no such expression, and the bitwise diffusion would make the combination of algebraic expressions difficult anyway.

Slide attacks. The sequence of round keys of SEA$_{n,b}$ is the same as the one of ICEBERG. Therefore the analysis done in [30] is still valid. Namely, the non-periodicity of the sequence should make slide attacks [11, 12] irrelevant. The particular structure of this sequence also has some similarities with the one of GOST, of which the vulnerability against slide attacks is examined in [12]. None of the attacks presented in [12] seems to be applicable to our cipher.

Related-key attacks. The first related-key attack has been described in [5]. It is the related-key counterpart of the slide attack. Such an attack is applicable when a round key K_i is computed from the previous round key K_{i-1} using a function f which is always the same: $K_i = f(K_{i-1})$. However in the case of SEA$_{n,b}$, a round constant that changes for each key round is used, which prevents this attack. Another type of related-key attack is the differential related-key attack [23, 24]. The non-linearity of the SEA$_{n,b}$ key schedule should prevent it. Moreover, note that the improvement of the differential related-key attack over classical differential cryptanalysis usually results from the fact that choosing a given round key difference allows to "counter" the effect of the diffusion layer on the differential characteristic; a typical example is the attack on 3-WAY [24]. As the security of SEA$_{n,b}$ against differential cryptanalysis results from its large number of rounds rather than from its diffusion, this effect is not relevant here.

Complementation properties. The DES has the following complementation property: if $P \xrightarrow{K} C$ denotes the fact that encryption of P under key K gives ciphertext C, then: $P \xrightarrow{K} C \Longleftrightarrow \overline{P} \xrightarrow{\overline{K}} \overline{C}$. The non-linear key scheduling and the presence of carry propagations in the actual SEA$_{n,b}$ algorithm prevents this property. We are not aware of any other similar structural feature in the design.

Algebraic attacks. Algebraic attacks intend to exploit the simple algebraic structure of a block cipher. For example, certain block ciphers can be written as an overdefined system of quadratic equations. Reference [13] argues that a method called XSL might provide a way to effectively solve this type of equations and recover the key from a few plaintext-ciphertext pairs. Clearly, SEA$_{n,b}$ has a simple algebraic structure, as it is based on a 3-bit substitution box. Therefore, if such an attack practically applies to a cipher like Serpent [1], it is likely applicable to one of the versions of our routines. As the complexity of XSL is supposedly polynomial in the plaintext size and number of rounds, it is specially true when those values increase. However, as the criteria for these techniques to be successful are still being discussed [9], we did consider this latter point as a scope for further research. We note that resistance against algebraic attacks would anyway exclude the use of small substitution boxes and therefore the possibility to build very low cost encryption routines.

3.3 Suggested number of rounds

From the previous descriptions, the minimum required number of rounds to provide security against known attacks would be $\frac{3n}{4} + 2 \cdot (n_b + \lfloor b/2 \rfloor)$. This roughly corresponds to the number of rounds to resist linear/differential attacks plus twice the number of rounds to obtain complete diffusion (to prevent both structural attacks and outer rounds improvements of statistical attacks). A more conservative approach (applied in most present block ciphers) would be to take a large security margin, *e.g.* by doubling this number of rounds[4]. n_r has to be odd: we add one if it is even. We also assume a minimum word size $b \geq 8$ bits.

4 Performance analysis

SEA$_{n,b}$ is targeted for being implemented on low-cost processors, with little code size and a small instruction set. However, SEA$_{n,b}$'s simple structure makes it easy to implement on any processor. In appendix, we propose a pseudo-assembly code of an encryption/decryption design with "on the fly" key scheduling. The implementation objectives were, in decreasing order of importance: (1) low RAM and registers usage, (2) low code size and (3) speed. It is based on the following (very) reduced instruction set (assuming 2-operand instructions only):

- Arithmetic and logic operators: $\vee, \wedge, \oplus, \boxplus, \ggg, \lll$.
- Branch instructions: goto, subroutine call and return.
- Comparison, load RAM in register, store register in RAM.

According to the code in appendix, the performances can be roughly estimated as follows. First, the combined number of RAM words and registers equals $5n_b + 3$. Then, the code size and implementation time (both in expressed in ops.) is evaluated by summing the values given in appendix. For the code size, it directly yields $31n_b + 36$ ops. For the implementation time, the round and key round respectively require $12n_b + 11$ ops. and $10n_b + 11$ ops. It yields a total of $(n_r - 1) \times (12n_b + 11 + 10n_b + 11 + 7) + (12n_b + 11) + 8n_b + 7$. These values are summarized in Table 1. Remark that, due to the particular structure of the key scheduling, we do not need to keep the master key in memory as, at the end of an encryption/decryption, we have $K_{n_r-1} = K_0$. Remark also that this implementation uses a low number of registers, namely $n_b + 3$. However, if more registers are available, they can be traded for RAM words, which will result in lower code size and faster implementation.

	# ram	# regs.	code size (ops.)	implementation time (ops.)
SEA$_{n,b}$	$4n_b$	$n_b + 3$	$31n_b$+36	$(n_r - 1) \times (22n_b + 29) + 20n_b + 18$

Table 1. Performance evaluation of SEA$_{n,b}$ (encryption + decryption).

[4] Note that the additional non-linearity provided by the modular addition also provides a security margin, under-estimated in our predictions.

For illustration purposes, we implemented SEA$_{n,b}$ on Atmel AVR ATtiny [3] and ARM [2] microprocessors. The Atmel ATtiny represents a typical target for such a low-cost encryption routine. We chose the ARM platform in order to provide rough comparisons between SEA$_{n,b}$ and the AES Rijndael.

Algorithm	E/D	Device	# ram	# regs.	code size	# clock cycles	# cycles × code size
SEA$_{96,8}$	yes	Atmel ATtiny	1	32	386	17 745	6849.10^3
SEA$_{192,32}$	yes	ARM (risc-32)	6	12	420	27 059	11 364.10^3
Rijndael [19]	no	ARM (risc-32)	16	12	1404	2889	4056.10^3
SEA$_{128,32}$	yes	ARM (risc-32)	6	12	280	18 039	5050.10^3

Table 2. Comparisons: the code size is expressed in bytes. The results of SEA$_{128,32}$ where obtained by multiplying the code size and number of cycles of SEA$_{192,32}$ by 2/3, since 128 is not a multiple of 6.

While direct comparisons are made difficult by their high dependencies on the target devices, the following general comments can be made:

- SEA$_{n,b}$ designs combine encryption and decryption more efficiently than most other encryption algorithms. In particular, key agility in decryption is usually not possible (*e.g.* for the AES Rijndael).
- The combined number of RAM words and registers of SEA$_{n,b}$ implementations (*i.e.* $5n_b + 3$) is generally lower than for other block ciphers.
- The code size of SEA$_{n,b}$ is generally lower than for other block ciphers implemented on similar platforms.

The flexibility of SEA$_{n,b}$ also makes it less sensitive to the choice of a processor than fixed-sized algorithms, although it is obvious that large buses improve efficiency. The drawback of these limited resources is in the number of cycles required for the encryption (*i.e.* SEA$_{n,b}$ trades space for time, which may be relevant due to present processors speeds). Looking at the code size - cycles product, the efficiency of SEA$_{n,b}$ remains similar to the one of Rijndael (encryption only) that is well known for its efficient smart cards implementations.

5 Conclusion

SEA$_{n,b}$ is a scalable encryption algorithm targeted for small embedded applications. The plaintext size, key size and processor (or word) size are parameters of the design. The structure of SEA$_{n,b}$ allows a fast evaluation of the cipher efficiency on any RISC machine. Its typical performances (encryption + decryption) for present key sizes and processors (*e.g.* 128-bit key, 1 Mhz 8-bit RISC) are in the range of an encryption/decryption in a few milliseconds, using a few hundreds bytes of ROM. One additional advantage of the design is its extreme simplicity. Based on the pseudo code provided in this paper, it is expected that the implementation of the cipher in assembly can be done within a few hours. We note finally that the design criteria of SEA$_{n,b}$ do not make it a conservative algorithm by nature. Further cryptanalysis efforts are consequently required.

Acknowledgements: The authors would like to thank François Koeune for his help and comments about ARM assembly tools and the NSF grant CCR-0122419, Center for Bits and Atoms. François-Xavier Standaert is a post doctoral researcher funded by the FNRS (Funds for National Scientific Research, Belgium).

References

1. R. Anderson, E. Biham, L. Knudsen, *Serpent: A Flexible Block Cipher With Maximum Assurance*, in the proceedings of The First Advanced Encryption Standard Candidate Conference, Ventura, California, USA, August 1998.
2. ARM, *32-bit RISC microprocessors*, http://www.arm.com/products/CPUs/
3. Atmel, *AVR 8-Bit RISC*, http://www.atmel.com/products/AVR/
4. E. Biham, A. Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, 1993, Springer Verlag.
5. E. Biham, *New types of cryptanalytic attacks using related keys*, Journal of Cryptology, vol 7, num 4, pp 229-246, Fall 1994, Springer Verlag.
6. E. Biham, A. Biryukov, A. Shamir, *Miss-in-the-Middle Attacks on IDEA, Khufu, and Khafre*, in the proceedings of FSE 1999, Lecture Notes in Computer Sciences, vol 1636, pp 124-138, Rome, Italy, March 1999, Springer-Verlag.
7. E. Biham, A. Biryukov, A. Shamir, *Cryptanalysis of Skipjack Reduced to 31 Rounds using Impossible Differentials*, in the proceedings of Eurocrypt 1999, Lecture Notes in Computer Sciences, vol 1592, pp 12-23, Prague, Czech Republic, May 1999, Springer Verlag.
8. E. Biham, O. Dunkelman, N. Keller, *The Rectangle Attack, Rectangling the Serpent*, in the proceedings of Eurocrypt 2001, Lecture Notes in Computer Science, vol 2045, pp 340-357, Innsbruck, Austria, May, 2001 Springer-Verlag.
9. A. Biryukov, C. De Cannière, *Block Ciphers and Systems of Quadratic Equations*, in the proceedings of FSE 2003, Lecture Notes in Computer Science, vol 2887, pp 274-289, Lund, Sweden, February 2003, Springer-Verlag.
10. A. Biryukov, C. De Cannière, M. Quisquater, *On Multiple Linear Approximations*, in the proceedings of Crypto 2004, Lecture Notes in Computer Science, vol 3152, pp 1-22, Santa Barbara, USA, August 2004, Springer-Verlag.
11. A. Biryukov, D. Wagner, *Slide attacks*, in the proceedings of FSE 1999, Lecture Notes in Computer Sciences, vol 1636, pp 245-259, Rome, Italy, March 1999, Springer-Verlag.
12. A. Biryukov, D. Wagner, *Advanced Slide Attacks*, in the proceedings of Eurocrypt 2000, Lecture Notes in Computer Science, vol 1807, pp 589-606, Bruges, Belgium, May 2000, Springer-Verlag.
13. N. Courtois, J. Pieprzyk, *Cryptanalysis of Block Ciphers with Overdefined Systems of Equations*, in the proceedings of Asiacrypt 2002, Lecture Notes in Computer Science, vol 2501 , pp 267-287, Queenstown, New Zealand, December 2002, Springer-Verlag.
14. N. Courtois, *Feistel Schemes and Bi-linear Cryptanalysis*, in the proceedings of Crypto 2004, Lecture Notes in Computer Science, vol 3152, pp 23-40, Santa Barbara, USA, August 2004, Springer-Verlag.
15. J. Daemen, R. Govaerts, J. Vandewalle, *A New Approach Towards Block Cipher Design*, in the proceedings of FSE 1993, Lecture Notes in Computer Science, vol 809, pp 18-32, Cambridge, UK, December 1993, Springer-Verlag.
16. J. Daemen, L. Knudsen, V. Rijmen, *The Block Cipher SQUARE*, in the proceedings of FSE 1997, Lecture Notes in Computer Science, vol 1267, pp 149-165, Haifa, Isreal, January 1997, Springer-Verlag.

17. J. Daemen, V. Rijmen, *The Design of Rijndael,* Springer-Verlag, 2001.
18. FIPS 197, *"Advanced Encryption Standard,"* Federal Information Processing Standard, NIST, U.S. Dept. of Commerce, November 26, 2001.
19. G. Hachez, F. Koeune, J.-J. Quisquater, *cAESar Results: Implementation of Four AES Candidates on Two Smart Cards*, in the proceedings of the Second Advanced Encryption Standard Candidate Conference, pp 95-108, Rome, Italy, March 1999.
20. N. Gershenfeld, R. Krikorian, D. Cohen, *The Internet of Things*, Scientific American, Octobre 2004, pp 76-81.
21. T. Jakobsen, L.R. Knudsen, *The Interpolation Attack on Block Ciphers*, in the proceedings of FSE 1997, Lecture Notes in Computer Science, vol 1267, pp 28-40, Haifa, Israel, January 1997, Springer-Verlag.
22. B.S. Kaliski, M.J.B. Robshaw, *Linear Cryptanalysis using Multiple Approximations*, in the proceedings of Crypto 1994, Lecture Notes in Computer Science, vol 839, pp 26-39, Santa Barbara, California, USA, August 1994, Springer-Verlag.
23. J. Kelsey, B. Schneier, D. Wagner. *Key-Schedule Cryptanalysis of IDEA, G-DES, GOST, SAFER, and Triple-DES*, in the proceedings of Crypto 1996, Lecture Notes in Computer Science, vol 1109, pp 237-251, Santa Barbara, California, USA, August 1996, Springer-Verlag.
24. J. Kelsey, B. Schneier, D. Wagner, *Related-Key Cryptanalysis of 3-WAY, Biham-DES, CAST, DES-X, NewDES, RC2, and TEA*, in the proceedings of ICICS 1997, Lecture Notes in Computer Sciences, vol 1334, pp 233-246, Bejing, China, November 1997, Springer-Verlag.
25. L.R. Knudsen, *Truncated and Higher Order Differentials*, in the proceedings of FSE 1995, Lecture Notes in Computer Sciences, vol 1008, pp 196-211, Leuven, Belgium, 1995, Springer-Verlag.
26. L.R. Knudsen and M.J.B. Robshaw, *Non-Linear Approximations in Linear Cryptanalysis*, in the proceedings of Eurocrypt 1996, Lecture Notes in Computer Science, vol 1070, pp 224-236, Saragossa, Spain, May 1996, Springer-Verlag.
27. S. Langford, M. Hellman, *Differential-Linear Cryptanalysis*, in the proceedings of Crypto 1994, Lecture Notes in Computer Science, vol 839, pp 17-25, Santa Barbara, California, USA, August 1994, Springer-Verlag.
28. M. Matsui, *Linear Cryptanalysis Method for DES Cipher*, in the proceedings of Eurocrypt 1993, Lecture Notes in Computer Science, vol 765, pp 386-397, Lofthus, Norway, May 1993, Springer-Verlag.
29. M. Matsui, *Supporting Document of MISTY1*, Submission to the NESSIE project, available from http://www.cosic.esat.kuleuven.ac.be/nessie/
30. F.-X. Standaert, G. Piret, G. Rouvroy, J.-J. Quisquater, J.-D. Legat, *ICEBERG : an Involutional Cipher Efficient for Block Encryption in Reconfigurable Hardware*, in the proceedings of FSE 2004, Lecture Notes in Computer Science, vol 3017, pp 279-299, New Delhi, India, February 2004, Springer-Verlag.
31. D. Wagner, *The Boomerang Attack*, in the proceedings of FSE 1999, Lecture Notes in Computer Sciences, vol 1636, pp 156-170, Rome, Italy, March 1999, Springer-Verlag.
32. D.J. Wheeler, R. Needham, *TEA, a Tiny Encryption Algorithm*, in the proceedings of FSE 1994, Lecture Notes in Computer Science, vol 1008, pp 363-366, Leuven, Belgium, December 1994, Springer-Verlag.
33. G. Yuval, *Reinventing the Travois: Encryption/MAC in 30 ROM Bytes*, in the proceedings of FSE 1997, Lecture Notes in Computer Science, vol 1267, pp 205-209, Haifa, Israel, January 1997, Springer-Verlag.

```
         Pseudo-assembly code:                              # ram       # regs.        # ops.

         % Init
         $L_0, R_0, KL_0, KR_0$ stored in RAM;              $4n_b$
         Set $i = 1$;                                                   1
         Set E/D;                                                       1

         % Subroutines (including return):
     $S$: $reg \leftarrow S(reg)$;                                      $n_b + 1$      $3n_b$+1
     $r$: $reg \leftarrow r(reg)$;                                      $n_b$          $n_b + 1$
    $sw$: switch $KL_i, KR_i$;                                          2              $4n_b + 1$

         Round:
         $reg \leftarrow R_i$;                                          $n_b$          $n_b$
         if $i \leq \lceil n_r/2 \rceil$                                               1
                   goto a:                                                             1
         $reg \leftarrow reg \boxplus KL_i$;                            $n_b + 1$      $2n_b$
         goto b:                                                                       1
      a: $reg \leftarrow reg \boxplus KR_i$;                            $n_b + 1$      $2n_b$
      b: call $S$;                                                                     1
         call $r$;                                                                     1
         if E/D=1;                                                                     1
                   goto c:                                                             1
         $reg \leftarrow reg \oplus L_i$;                               $n_b + 1$      $2n_b$
         goto d:                                                                       1
      c: $reg \leftarrow reg \oplus R(L_i)$;                            $n_b + 1$      $2n_b$
      d: $L_{i+1} \leftarrow R_i$;                                      1              $2n_b$
         if E/D=1;                                                                     1
                   goto e:                                                             1
         $R_{i+1} \leftarrow R^{-1}(reg)$;                              $n_b$          $n_b$
         goto f:                                                                       1
      e: $R_{i+1} \leftarrow reg$;                                      $n_b$          $n_b$
      f: return;                                                                       1

         Key round:
         $reg \leftarrow KR_i$;                                         $n_b$          $n_b$
         if $i < \lceil n_r/2 \rceil$                                                  1
                   goto g:                                                             1
         $temp \leftarrow n_r - i$;                                     1              2
         $reg \leftarrow reg \boxplus temp$;                            $n_b + 1$      1
         goto h:                                                                       1
      g: $reg \leftarrow reg \boxplus i$;                               $n_b$          1
      h: call $S$;                                                                     1
         call $r$;                                                                     1
         $reg \leftarrow R(reg) \oplus KL_i$;                           $n_b + 1$      $2n_b + 1$
         $KL_{i+1} \leftarrow KR_i$;                                    1              $2n_b$
         $KR_{i+1} \leftarrow reg$;                                     $n_b$          $n_b$
         return;                                                                       1

         % Total:
      j: call round;                                                                   1
         if $i \neq \lceil n_r/2 \rceil$                                               1
                   goto k:                                                             1
         call $sw$;                                                                    1
      k: if $i = n_r$                                                                  1
                   goto end:                                                           1
         call key round;                                                               1
         $i = i + 1$;                                                                  1
         goto j:                                                                       1
    end: call $sw$;                                                                    1
         switch $L_i, R_i$;                                             2              $4n_b$
```