Sea Monkey CryptoSystem

CS13B027 Hemanth Kumar Tirupati
CS13B046 Aravind Krishna

March 1, 2016

Abstract

We propose Sailing Sea Monkey, a 32-bit block extension to AES-128 offering same security as AES-128 and also more efficiency on 32 bit architectures. Firstly we prove security against various cyptanalytic techinques known against block ciphers *viz* Linear Cryptanalysis and Differential Cyptanalysis. After proving the computational security of the cipher, we discuss implementation aspects that become liability for secure-ness of the cipher *viz side channel attacks.* We do so by demonstrating the security of an implementation against Timing Analysis and BufferOverflow attack[1].

1. Sailing Sea Monkey is a variant of AES for 32 bit blocks and uses a fixed key length of 128 bits. Even though key length is fixed at 128 bits, it can be extended to higher key lengths.

2. We apply same endomorphic system in each of 10 iterations. So, number of *rounds* 10. Details of using these many rounds is explained in Q4.

3. Each round of cipher consists of

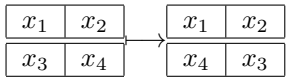
   Subbytes : Consists of replacing each byte of plaintext block with corresponding inverse in $\mathcal{F}_{2^8}$ and scrambling. The idea

   scrambling is to prevent algebraic attacks on cipher.

   This is the only non linear layer of the round. Use of this operation is to induce non linearity into our cryptosystem which will increase the security if applied iteratively.

   $Subbytes(x) \triangleq A(x^{-1}) \oplus b$.

   Shiftrows : Consists of left shifting the second row of bytes.

   The purpose of this round is induce $confusion$ in our cryptosystem, which hides the relation between cipher text and plaintext i.e. it provides illusion that

   $Pr[Y = y \mid X = x] \approx Pr[X = x]$ (*ideal secrecy criteria)* i.e. there is very small correlation between $\mathcal{C}$ and $\mathcal{P}$.

   In other words it eliminates any bias that certain keys produce certain cipher texts more frequently.

   MixColumns : Consists of multiplication with a maximum distance separable matrix. This step is responsible for achieving diffusion.

   The purpose of this operation is to induce hide the correlation between keys and cipher texts .i.e, it provides illusion of ideal secrecy condition being satisfied

   $Pr[Y = y \mid K = k] \approx Pr[K = k]$.

   In other words it eliminates any bias that certain keys produce certain cipher texts more frequently.

   AddroundKey : We Ex-Or the output of above steps with a psuedo random round key generated from the master key.

   This is the only part of encryption algorithm which is kept hidden from the adversary.

4. Number of rounds have to be fixed so that

[1] Part of next paper

- It is immune to Known Plain Text Attack :
  - number of round key guesses $\geq 2^{90}$(assumed to be infeasible in current day computational power) so that it is infeasible for adversary to guess the round keys and decrypt the cipher. Since each round uses 2^{32}bit round keys, the number of rounds must be atleast 3.
- It is immune to Chosen Plain Text Attack :
  - Square is one such attack which is shown to be computationally feasible if number of rounds $\leq$6.
- However 6 is not a good choice for practical implementation since we don't know if there might some other Chosen Plain Text Attack that can exploit AES in better way that Square Attack. Hence we erred on side of caution fix number of rounds at 10 to provide security at the expense of efficiency.

5. S-Boxes are straight and have size of 8×8.

   - S-Box size has been chosen as 8×8, reason being bigger S-Boxes provide higher non linearity. This implies more resistance against Linear and Differential Cryptanalysis.
     - 16×16 would not be a good design choice since we would have to work in $\mathcal{F}_{2^{16}}$field and must store $4GB$ worth of tables. In this if we chose compute the inverses on the fly instead of look up table, it might induce a timing side channel which compromises security.
     - Other numbers $n \times n \; \exists n \neq 2^k$ would induce efficiency overhead on the algorithm. Hence the choice of 8×8.
   - S-boxes are straight:
     - Using expansion S-Boxes doesn't improve security since it is *pseudo* random transformation of original plain text. However it would add to more confusion and diffusion. We ignored this advantage for efficiency.
     - Using compression S-Box is ruled out since our algorithm is based on Substitution Permutation Network.

6. S-box design must be done so that they have

   - Strict Avalanche Effect
   - Balancedness
   - Nonlinearity

   Inverse of x in $\mathcal{F}_{2^8}$ field can be proven (refer Q7) to have satisfy afore mentioned properties. Hence the choice.

7. We automated the verification of various desirable S-box properties (corresponding sources are attached). The following are observations

   (a) They are balanced. This can be verified by checking number of times each bit is set in the output of S-box for all possible inputs. Number of ones toggled at any of eight positions is 128 $\implies$ our S-Boxes are balanced.

   ```
   $g++ -std=c++11 balanced.cpp
   $./a.out
   {128, 128, 128, 128, 128, 128, 128, 128}
   ```

   (b) SAC can be verified by checking $f(x \oplus \alpha) \oplus f(x)$ is balanced where exactly one bit of α is toggled. Our S-boxes do not satisfy strict avalanche criteria.

   ```
   $g++ -std=c++11 sac.cpp
   $./a.out
   {132, 132, 116, 144, 116, 124, 116, 128}
   {120, 124, 144, 128, 124, 116, 128, 136}
   {132, 132, 128, 120, 144, 128, 136, 128}
   {136, 136, 120, 116, 128, 136, 128, 140}
   {116, 128, 116, 132, 128, 128, 140, 136}
   {116, 132, 132, 120, 120, 140, 136, 136}
   {136, 136, 120, 132, 120, 136, 136, 124}
   {132, 144, 132, 136, 124, 136, 124, 132}
   ```

(c) Non linearity can be verified Hadamand Matrix. Minimum non linearity of the *any* bit is 112.

```
$g++ -std=c++11 nonlinearity.cpp
$./a.out
112
```

It has been shown that it is impossible to have function that is both SAC and Bent. AES S-Box offers a balance between both.

8. Linear Approximation table is a 256 × 256 table which can be generated using $\$./licran.$

9. Differential distribution table is 256 × 256 table which can be generated using $\$./dcran.$

10. For a matrix M to be Maximum Distance seprable, $\widetilde{M} = \begin{bmatrix} I_{n\times n} \\ M_{m\times n} \end{bmatrix}$. M will be MDS $\iff$ all matrices obtained by removing any m rows of $\widetilde{M}$ should be non singular[2].

$$\widetilde{M} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ a & b \\ c & d \end{bmatrix}$$

By using above property, we get following relation between a, b, c, d.

$a \neq 0, b \neq 0, c \neq 0, d \neq 0$ and $ad \neq bc$.

So, we choose $M = \begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} 1 & 2 \\ 1 & 3 \end{bmatrix}$ that satisfy above relations. It also has inverse in $\mathcal{F}_{2^8}$.

$M^{-1} = \begin{bmatrix} 3 & 2 \\ 1 & 1 \end{bmatrix}$.

11. It takes 2 rounds for achieving complete diffusion.

Proof:

Let $P = \begin{array}{|c|c|}\hline x_1 & x_2 \\ \hline x_3 & x_4 \\ \hline \end{array}$ be representation of plaintext.

Upon changing one bit in x_1,say we get x_1^*.

$P^* = \begin{array}{|c|c|}\hline x_1^* & x_2 \\ \hline x_3 & x_4 \\ \hline \end{array}$

So, P^* on Subbytes, we get same output as P, except that the first byte differs.

$P_{sub} = Subbytes(P) = \begin{array}{|c|c|}\hline y_1 & y_2 \\ \hline y_3 & y_4 \\ \hline \end{array}$

$P^*_{sub} = Subbytes(P^*) = \begin{array}{|c|c|}\hline y_1^* & y_2 \\ \hline y_3 & y_4 \\ \hline \end{array}$

After Shiftrows,

$P_{shift} = Shiftrows(P_{sub}) = \begin{array}{|c|c|}\hline y_1 & y_2 \\ \hline y_4 & y_3 \\ \hline \end{array}$

$P^*_{shift} = Shiftrows(P^*_{sub}) = \begin{array}{|c|c|}\hline y_1^* & y_2 \\ \hline y_4 & y_3 \\ \hline \end{array}$

Upon multiplication with M, we get the disturbed bits into the whole first row since M is has property of being MDS.

$P_{mix} = MixColumns(P_{shift}) = \begin{array}{|c|c|}\hline z_1 & z_2 \\ \hline z_4 & z_3 \\ \hline \end{array}$

$P^*_{mix} = MixColumns(P^*_{shift}) = \begin{array}{|c|c|}\hline z_1^* & z_2^* \\ \hline z_4 & z_3 \\ \hline \end{array}$

AddRound key doesn't any more disturbance.

[2]Source: Wiki

$P_{add} = AddroundKey(P_{mix}) =$ $\begin{array}{|c|c|}\hline w_1 & w_2 \\ \hline w_4 & w_3 \\ \hline \end{array}$

$P^*_{add} = AddroundKey(P^*_{mix}) =$ $\begin{array}{|c|c|}\hline w^*_1 & w^*_2 \\ \hline w_4 & w_3 \\ \hline \end{array}$

So, after one round, the output of P and P^*differ in first two bytes.

The main observation here is only Mixcolumns adds to disturbance. At the beginning of second round, P and P^*differ in a single row *viz* first two bytes. After Subbytes and Shiftrows also they differ only in first two bytes. Now, by property of MDS matrix the disturbance propagates to all the bytes.

Let the ecrypted text after *Subbytes* of second round be for the above plain texts

$C =$ $\begin{array}{|c|c|}\hline a_1 & a_2 \\ \hline a_3 & a_4 \\ \hline \end{array}$ and $C^* =$ $\begin{array}{|c|c|}\hline a^*_1 & a^*_2 \\ \hline a_4 & a_3 \\ \hline \end{array}$

$\implies$ upon multiplication with, $M = \begin{bmatrix} 1 & 2 \\ 1 & 3 \end{bmatrix}$, we get $M \times C$ and $M \times C^*$.

$M \times C = \begin{bmatrix} 1 & 2 \\ 1 & 3 \end{bmatrix} \times \begin{bmatrix} a_1 & a_2 \\ a_3 & a_4 \end{bmatrix}$ and $M \times C^* = \begin{bmatrix} 1 & 2 \\ 1 & 3 \end{bmatrix} \times \begin{bmatrix} a^*_1 & a^*_2 \\ a_3 & a_4 \end{bmatrix}$ $\implies$ Both differ in all the elements .i.e we achieved complete diffusion after 2^{nd} rounds *MixColumns*. So, minimum number of rounds required for complete diffusion = 2.

Linear and Differential Cyptanalysis of Sailing Sea Monkey

CS13B027 Hemanth Kumar Tirupati

CS13B046 Aravind Krishna

CS13B062 Shreyas Harish

March 10, 2016

Abstract

In this paper we give formal mathematical proof for the resistance of Sailing Sea Monkey against general attacks on block ciphers *viz* Linear and Differential Cryptanalysis. We provide a lower bound on minimum number of Plain Text and Cipher Text pairs needed for cracking the proposed cipher with *sufficient* confidence. These lower bounds turn out to be much higher than number of key verfications (i.e, 2^{128})required by brute force and therefore are not viable attacks on the cipher.$\diamondsuit$

1 Introduction

Let the input to our cipher be $\boxed{a}\boxed{b}\boxed{c}\boxed{d}$

Represented in matrix form, $Input = \begin{bmatrix} a & c \\ b & d \end{bmatrix}$

The output of Sbox be

$S = SBox(Input) = \begin{bmatrix} A & C \\ B & D \end{bmatrix}$

After SBox layer, we perform shift-rows operations and multiplication with MDS matrix $\begin{bmatrix} 1 & 2 \\ 1 & 3 \end{bmatrix}$.

Therefore final output of the first layer would be $O_1 = \begin{bmatrix} A+2D & B+2C \\ A+3D & B+3C \end{bmatrix}$

$\implies O_1 = \boxed{A+2D}\boxed{A+3D}\boxed{B+2C}\boxed{B+3C}$

$\implies$ Disturbing one bit of input (say A) disturbs first two bytes input to next round.

Additionally it can also be seen that disturbing first two bytes (A and B) of input to a round disturbs all the output bytes.

Following table summarizes the working of cipher.

	Round 1			
Input Plaintext	a	b	c	d
Key XORing and SBoxing	A	B	C	D
Shift Rows	A	D	C	B
Mix Columns	$A+2D$	$A+3D$	$B+2C$	$B+3C$

Output of *Round 1* would be as input to passed to *Round 2*

	Round 2			
Round 2 input	$A+2D$	$A+3D$	$B+2C$	$B+3C$
Key XORing and SBoxing	$f_1(A,D)=X$	$f_2(A,D)=Y$	$f_3(B,C)=Z$	$f_4(B,C)=W$
Shift Rows	X	Z	W	Y
Mix Columns	$X+2Z$	$X+3Z$	$Y+2W$	$Y+3W$

2 Resistance to Linear Cryptanalysis

Proposition 1. *Total Bias of linear approximation involving n SBoxes is decreasing function in n.*

Inorder to maximize bias of a linear combination, the adversary must choose a path such that total bias of linear approximation (involving n SBoxes) $\epsilon = 2^{n-1}\epsilon_1\epsilon_2...\epsilon_n$ is maximized.

$\implies \epsilon \leq 2^{n-1}\epsilon_0^n$ where ϵ_0 is maximum bias possible for any linear approximation of SBox.

$\implies$ $2^{n-1}\epsilon_0^n$ is the maximum bias that adversary can squeeze out of any linear approximation involving n SBoxes.

In our cipher $\epsilon_0 = 0.0625 \implies 2\epsilon_0 < 1 \implies$ total bias is decreasing function in terms of number of SBoxes involved.$\diamondsuit$

So, maximizing possible bias ,$2^{n-1}\epsilon_0^n$ is same as minimizing n, the number of SBoxes involved.

Proposition 2. *Any linear trail on the cipher involves a minimum of 31 SBoxes.*

Proof. In order to construct any linear trail □

- Choose bits of input plain text that are going to be part of final linear approximation
- Construct a linear trail by tracing through linear approximation of these bits to next round.

  This trace through invloves finding all active SBoxes that depend on output bits corresponding to

linear approximation involving the chosen bits.

Example. Say in input $x_1x_2.....x_{32}$, we tried to construct a linear trail involving x_3, x_4, x_5 and linear approximation we use is $x_3 \oplus x_4 \oplus x_5 \oplus y_1 = 0 \implies$ in further rounds, we must include all SBoxes that depend on $y_1$as part of linear approximation.

In our cipher, diffusion optimality occurs after 2 rounds of encryption i.e, after two rounds, all 4 SBoxes in a round become part of linear combination.

So, minimum number of SBoxes that would contribute to final bias $= \underbrace{1}_{round1} + \underbrace{2}_{round2} + \underbrace{4 \times 7}_{rounds\,3-9} =$ $31\diamondsuit$.

Using Propositions 1 and 2 we can say any linear approximation, adversary can get bias that is bounded above by

$$\epsilon_{threshold} = 2^{31-1} \times \epsilon_0^{31} \approx 5.05 \times 10^{-29}$$

Inorder to guess partial key bits of penultimate round, we need $\mathcal{O}(\frac{1}{\epsilon^2})$ known plain-text cipher-text pairs[1] $\implies \geq 3.92 \times 10^{56}$KPT's are required to guess last round partial key bits.

So, a brute force verification of $2^{128} keys = 3.40 \times 10^{38}$ has lower complexity than linear cryptanalysis.

So, linear cryptanalysis is not a viable attack on Sea Monkey.

3 Resistance to Differential Cryptanalysis

Propositions 1 and *2* have similar extensions for differential cryptanalysis.

Proposition 3. *Total propagation ratio of differential involving n SBoxes is decreasing function in n.*

Proof. □

- Propagation ratio p for the differential trail is the product of the propagation ratio for each of the SBoxes involved in the trail.
- $\implies p = p_1p_2....p_n$ where p_i denotes propagation ratio of i^{th} SBox.
- In our cipher, maximum propagation ratio is bounded above by $0.015625 < 1. \implies p$ is decreasing function in number of SBoxes involved in differential trail.

Proposition 4. *Any differential trail on the cipher involves a minimum of 31 SBoxes.*

Proof. □

[1]From Matsui Linear Cryptanalysis

- This would involve a similar proof as the minimum number of SBoxes that get involved in any linear trail.
- Choose a set of input bits as the starting point of the differential trail.
- This would involve atleast one sbox in the first round.
- So, two SBoxes in round 2 would be part of the differential trail. For higher rounds, all *4* SBoxes would be part of the differential trail (reason explained above in *Proposition 2*).
- So, minimum number of SBoxes that would contribute to final propagation ratio

$$= \underbrace{1}_{round1} + \underbrace{2}_{round2} + \underbrace{4 \times 7}_{rounds\,3-9} = 31\diamondsuit.$$

As mentioned earlier, in case of differential cryptanalysis, the total propagation ratio involving n SBoxes is given by $p = p_1 p_2 ....p_n$.

This propagation will be bounded above by p_0^n where p_0 is maximum propagation value of any SBox.

Value of p_0 in our Sboxes is 0.015625.

By *Proposition 4*, n$\geq$31 $\implies$ $p \leq 1.02 \times 10^{-56}$.

$\therefore$ We would require $\mathcal{O}(\frac{1}{p}) \approx 9.80 \times 10^{55}$ chosen plain texts to find target partial key bits of penultimate round which is much higher than $2^{128} \approx 3.40 \times 10^{38}$ key verifications needed in case of a brute force attack.

Remark. So, we proved the security of Sailing Sea Monkey against Linear Cryptanalysis and Differential Cryptanalysis.$\diamondsuit$

Verification of ϵ_0 and p_0 values

Values of ϵ_0 and p_0 for SBoxes have been calculated by enumerating the Linear Approximation Table and Differential Distribution Table.

In order to verify these values use

```
$make maxbias
$make maxpropratio
$./maxbias
e = 0.062500000000000
$./maxpropratio
p = 0.015625000000000
```

Modifications to Previous Submission

We *are* not making any changes to our originally proposed encryption algorithm.

Software Implementation of Sailing Sea Monkey

CS13B027 Tirupati Hemanth Kumar
CS13B046 Aravind Krishna
CS13B062 Shreyas Harish

March 27, 2016

Abstract

In the final part of assignment we implement the afore-mentioned block cipher, optimized for *X86* architectures. The efficiency lies in the fact that entire program of encryption(15 KB) and decryption (15 KB) can entirely fit into L_1 caches of most modern day machines.

Implementational Aspects of Ciphers

The following are some software efficient techinques used for implementing the cipher algorithm.

- Size of Executable:
  - ▶ We have designed our implementation so that it can effectively fit into few blocks of *L1* cache.
  - ▶ Inorder to achieve this, we computed T-tables for the proposed MDS matrix. T-table construction for encryption and decryption are described below.
- Encryption T-table construction:
  - ▶ Since the MDS matrix, $M = \begin{pmatrix} 1 & 2 \\ 1 & 3 \end{pmatrix}$, the encryption T-table would contain $a\|d$ and $2 \times a\|3 \times d$ values.
  - ▶ However storing $a\|d$ T-table is redundant since there is no multiplication involved in this table(i.e, SBox outputs can be directly used). $\implies$ We only need to store a single T-table that contains $2 \times a\|3 \times d$ values which has size of 512 bytes.
- Decryption T-table construction:
  - ▶ Inverse MDS matrix is given by $M^{-1} = \begin{pmatrix} 3 & 2 \\ 1 & 1 \end{pmatrix}$ $\implies$ Decryption T-table must contain $3 \times a\|b$ and $2 \times a\|b$ values.
  - ▶ This would account for size of $512 \times 2\,bytes$. On observing that this is same as size of storing 2 multiplication tables ($256 \times 2\,bytes$) in AES_{128} field, we can see that T-tables have space overhead over storing multiplication tables. Hence, we directly used multiplication tables of the field instead of T-tables incase of decryption.

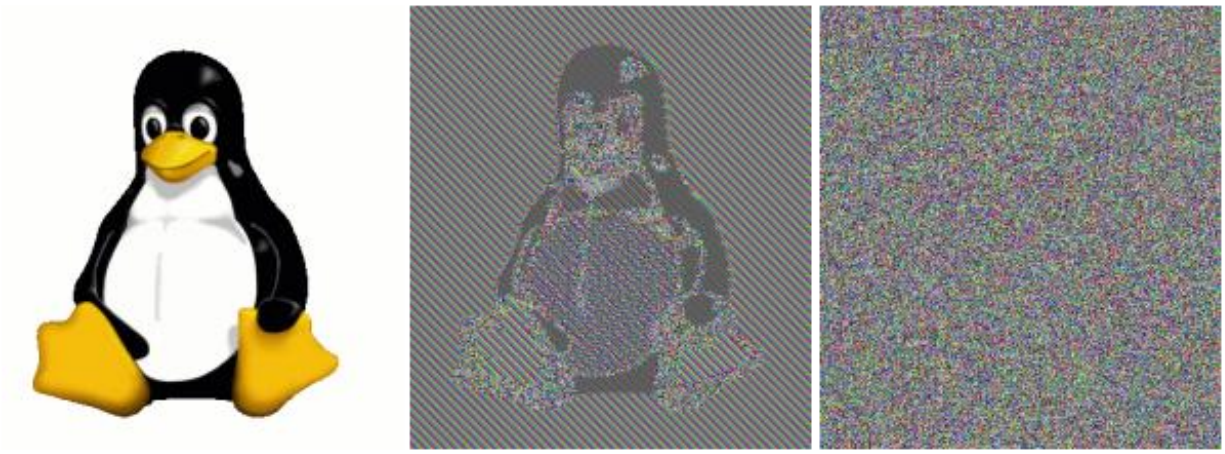


Figure 1: information leak in ECB vs CBC

- Mode Of Operation:
  - We use the cipher in Cipher Block Chain Mode so that low entropy information can also be sent without loss of security.
  - For every encryption a new IV is generated using master key.
  - Because of serious disadvantage of ECB i.e having the same cipher text block for same input block leads to various security attacks and also can leak information. This can be easily seen from the encryption of an image with ECB and CBC where ECB leaks information about the source image wheras CBC avoids this.
- We have also specifically avoided any key specific computations in the implementation so that the executable run time does not have any correlation with key used.
- We have used AES Key scheduling algorithm to generate round keys and hence haven't taken any chance in various security issues that might arise because of formally proved secure round key generation implementations.
  -

Changes to Previous Submission

While encryption algorithm itself is left unchanged, its mode of operation is fixed as CBC and we have proceeded abiding to all the specifications about the security and efficiency.

Encryption time vs size of file

The following graph depicts runtime of encryption and decryption on various file sizes.

We can see that run time of program is linear *w.r.t* number of bytes encrypted. This can attributed to CBC mode of operation which encrypts the message block by block.

Working of the Cipher

Working of cipher can verified under ascii and non-ascii inputs can be easily verified using

```
$ make enc                                  1
$ make encrypt                              2
$ make dec                                  3
$ make decrypt                              4
$ diff alice.txt alice.decrypt              5
$ make pic                                  6
$ make depic                                7
```

- 1 To compile encryption code:
- 2 To encrypt the alice.txt using aes.key and produces alice.encrypt
- 3 To compile decryption code:
- 4 To decrypt the alice.decrypt using aes.key and produces alice.decrypt
- 5 To show that our encryption and decryption functions properly for **non-ascii** inputs encrypt an image file pic.jpg
- 7 To decrypt the pic.encrypt produced use
- We get pic.decrypt and we can use diff of the pic.jpg and pic.decrypt or use an image viewer to see that both are indeed the same file.