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.

   $Subbytes(x) \overset{\Delta}{=} A(x^{-1}) \oplus b.$

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

   $$\begin{array}{|c|c|}\hline x_1 & x_2 \\ \hline x_3 & x_4 \\ \hline \end{array} \longmapsto \begin{array}{|c|c|}\hline x_1 & x_2 \\ \hline x_4 & x_3 \\ \hline \end{array}$$

   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

   $Pr[Y = y \,|\, X = x] \approx Pr[X = x]$ (*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

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

   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.