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.

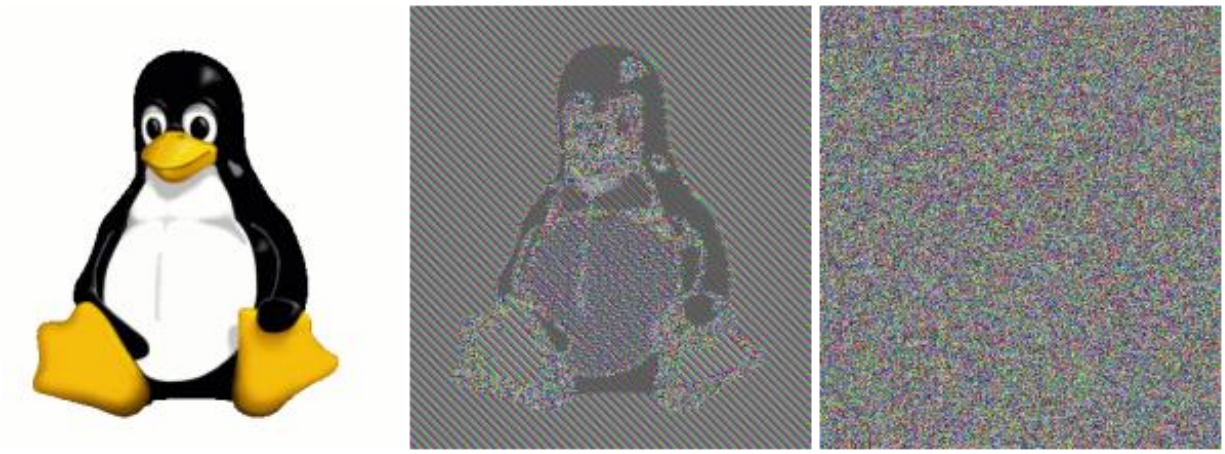

Figure 1: information leak in ECB vs CBC

- 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.