A Report on

Cryptoware: AES-192 (CFB Mode)

(Mini-Project of CSE1007-Introduction to Cryptography)

Submitted by

Aryan Subrahmonyaru

(Registration-19BCE7347)

on

15 NOV 2020

School of Computer Science and Engineering

VIT-AP University, Andhra Pradesh

Abstract

The Cipher Feedback (CFB) mode is a typical block cipher mode of operation using block cipher algorithm. In this version, we provide Data Encryption Standard (DES) and Advanced Encryption Standard (AES) processing ability, the cipherkey length for DES should be 64 bits, and 128/192/256 bits for AES. Another limitation is that our working mode works on units of a fixed size (64 or 128 bits for 1 block), but text in the real world has a variety of lengths. So, the last block of the text provided to this primitive must be padded to 128 bits before encryption or decryption. Although, CFB1 and CFB8 modes share the same interface with CFB128 mode, the plaintext and ciphertext is processed bit-by-bit or byte-by-byte not block-by-block for CFB1 and CFB8 modes respectively.

1 Introduction

The Advanced Encryption Standard (AES), was developed by Vincent Rijmen, Joan Daemen. The CFB (Cipher Feedb ack) mode of operation allows the block encryptor to be used as a stream cipher. It's a symmetric algorithm ,so it does not have public and private keys - only a shared secret.

2 About

The CFB mode of operation allows the block encryptor to be used as a stream cipher. It also needs an IV. First, CFB will encrypt the IV, then it will xor with plaintext block to get ciphertext. Then we will encrypt the encryption result to xor the plaintext.

3 Implementation environment

Programming language used for implementation is Java.

4 Procedure

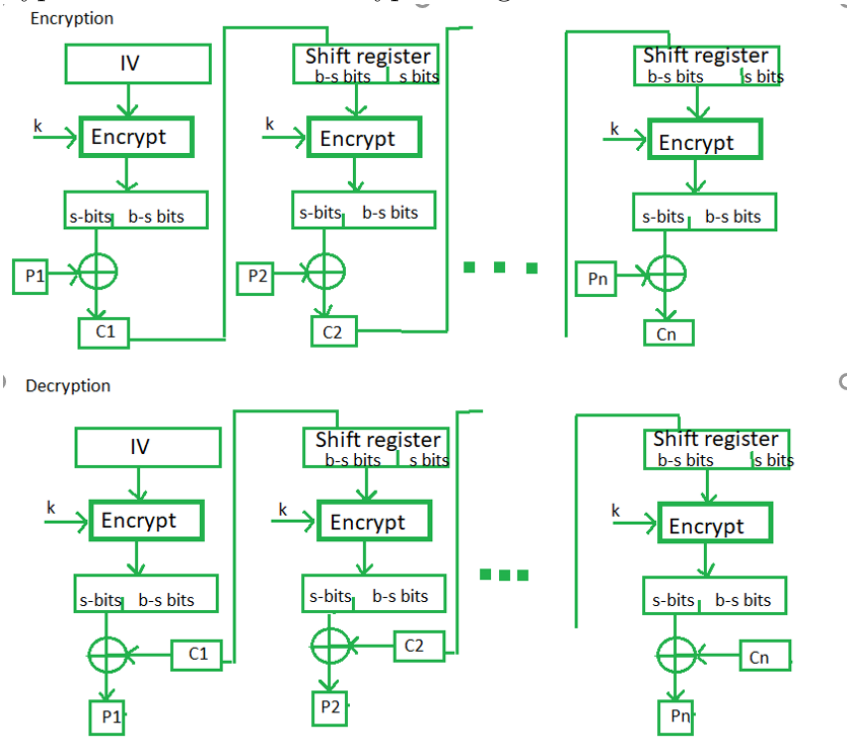
4.1 Encryption

The CFB (Cipher FeedBack) mode of operation allows the block encryptor to be used as a stream cipher. It also needs an IV.

First, CFB will encrypt the IV, then it will xor with plaintext block to get ciphertext. Then we will encrypt the encryption result to xor the plaintext. Because this mode will not encrypt plaintext directly, it just uses the ciphertext to xor with the plaintext to get the ciphertext. So in this mode, it doesn't need to pad data. And it could decrypt data in parallel, not encryption. This mode is similar to the CBC, so if there is a broken block, it will affect all following block. This mode can be attacked by replay attack. For example, if you use the other ciphertext to replace the new ciphertext, the user will get the wrong data. But he will not know the data is wrong. It is safe from CPA, but it is easily sysceptible to CCA. To ensure security, the key in this mode need to be changed for every 2Raise((n+1)/2) encryption blocks.

4.2 Decryption

Decryption uses the same encryption algorithm.



5 Major Components

Code for AES CFB-192

```
import java.security.MessageDigest;
import java.util.Arrays;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.crypto.spec.SecretKeySpec;
import javax.crypto.spec.IvParameterSpec;



import javax.crypto.Cipher;
import javax.crypto.spec.IvParameterSpec;

import javax.crypto.spec.SecretKeySpec;



public class AES {
static String IV = "AAAAAAAAAAAAAAAA";
static String plaintext = "test text 123\0\0\0";
static String encryptionKey = "0123456789abcdef";

public static void main(String [] args)
{
try {

System.out.println("==Java==");
System.out.println("plain:   " + plaintext);



byte[] cipher = encrypt(plaintext, encryptionKey);
```

```
System.out.print("cipher:  ");
for (int i=0; i<cipher.length; i++)
System.out.print(new Integer(cipher[i])+" ");
System.out.println("");


String decrypted = decrypt(cipher, encryptionKey);


System.out.println("decrypt: " + decrypted);


}
catch (Exception e)
{
     e.printStackTrace();
}
}


public static byte[] encrypt(String plainText, String encryptionKey) throws Exception
{
Cipher cipher = Cipher.getInstance("AES/CBC/NoPadding", "SunJCE");
SecretKeySpec key = new SecretKeySpec(encryptionKey.getBytes("UTF-8"), "AES");
cipher.init(Cipher.ENCRYPT_MODE, key,new IvParameterSpec(IV.getBytes("UTF-8")));
return cipher.doFinal(plainText.getBytes("UTF-8"));
}


public static String decrypt(byte[] cipherText, String encryptionKey) throws Exceptio
{
```

```
Cipher cipher = Cipher.getInstance("AES/C/NoPadding", "SunJCE");
SecretKeySpec key = new SecretKeySpec(encryptionKey.getBytes("UTF-8"), "AES");
cipher.init(Cipher.DECRYPT_MODE, key,new IvParameterSpec(IV.getBytes("UTF-8")));
return new String(cipher.doFinal(cipherText),"UTF-8");
}
}
```

6 Result

Result

```
$javac AES.java

$java -Xmx128M -Xms16M AES
==Java==
plain:   test text 123
cipher:  16 -124 41 -83 -16 -123 61 -64 -15 -74 87 28 63 30 64 78
decrypt: test text 123
```

7 Refrences

Refrences: [1] Behrouz A Forouzan, Debdeep Mukhopadhyay, “Cryptography and Network Se- curity”, Mc Graw Hill, Third Edition, 2015.

[2] William Stallings, “Cryptography and Network Security: Principles and Prac- tice”, Pearson Education, Seventh Edition, 2017.

[3] Overleaf: a collaborative cloud-based LaTeX editor used for writing, editing and publishing scientific documents.

http://www.overleaf.com.