

Assignment 2 Report

Block Cipher Modes

Wee JunJie
U1540273K

1 ECB Mode

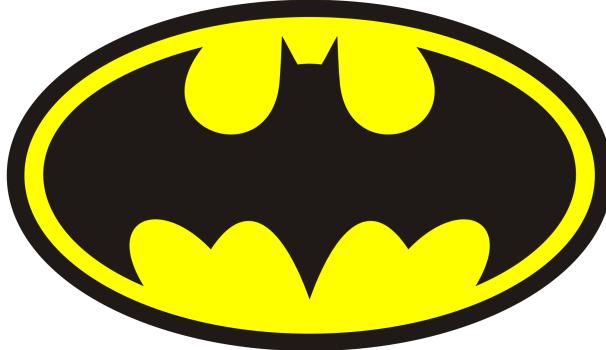
1.1 Implementation

The solution code (version 1) for Assignment 1 was being used and modified for Task 2 and Task 3 as well. In the `main.cpp`, ECB Mode can be selected by the user as Mode 1. In this report, all the encryptions and decryptions uses the key: 0f0e0d0c0b0a0908 0706050403020100.

1.2 Encryption of Bitmap File

1.2.1 Bitmap File Header

The Bitmap File Format contains the BMP File Header and the DIB Info Header. The BMP File Header is of 14 bytes and DIB Info Header is of 40 bytes. The first 4 bytes of the BMP File Header represents the file signature of Bitmap. i.e. 42 4D. The next 4 bytes shows the size of the respective Bitmap file. In this case, it shows 0E BF 26 00 in little endian hence representing 2,539,278 bytes in decimal notation. Following this, the subsequent 4 bytes are reserved for the application that created the Bitmap. The last 4 bytes of BMP File Header shows the starting address of the Bitmap Image Data. In this case, the starting address is at 36 as shown at byte address 0A in Figure 1. Hence, it can be seen starting at byte address 36 with a 17 as well. The DIB Info Header in general consists of the image information and the pixel format of the BMP. In total, 54 bytes needs to be copied in order to open in BMP File Format.



(a) Original Image

```

HxD - [C:\Users\J\Documents\GitHub\CryptoAssignment2\SpeckAI\O\batman-logo-big.bmp]
File Edit Search View Analysis Extras Window ?
16 ANSI hex
batman-logo-big.bmp

Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000000 42 4D 0E BF 26 00 00 00 00 00 36 04 00 00 28 00 BM.i6.....6...
00000010 00 00 18 08 00 00 C9 04 00 00 01 00 08 00 00 00 ....E.....
00000020 00 00 00 00 00 74 12 00 00 74 12 00 00 01 .....t.....
00000030 00 00 00 01 00 00 17 1A 1F FF 17 20 25 FF 2C 2E .....g. %y..
00000040 33 FF 17 36 3A FF 0E 6E 71 FF 13 4F 53 FF 6D 6F 3y.e;y.ngy.OSymo
00000050 72 FF 4F 52 55 FF 2B 7E 81 FF 0B 8F 92 FF 04 D0 ryORUy--.y..y.B
00000060 D1 FF 00 FF FF 00 F8 F0 02 E8 FF 00 AF Ny.yyy..wy..eey.
00000070 B1 FF 8F 8F FF D0 D1 FF FF FF FF F8 F8 2y.'ybDlyyyyyw
00000080 F8 FF E7 E8 FF AE AF B1 FF 00 00 FF 00 00 wycééý-2y...y..
00000090 00 FF 00 00 FF 00 00 FF 00 00 FF 00 00 .y...y...y...y..
000000A0 00 FF 00 00 FF 00 00 FF 00 00 FF 00 00 .y...y...y...y..
000000B0 00 FF 00 00 FF 00 00 FF 00 00 FF 00 00 .y...y...y...y..
000000C0 00 FF 00 00 FF 00 00 FF 00 00 FF 00 00 .y...y...y...y..
000000D0 00 FF 00 00 FF 00 00 FF 00 00 FF 00 00 .y...y...y...y..
000000E0 00 FF 00 00 FF 00 00 FF 00 00 FF 00 00 .y...y...y...y..
000000F0 00 FF 00 00 FF 00 00 FF 00 00 FF 00 00 .y...y...y...y..
00000100 00 FF 00 00 FF 00 00 FF 00 00 FF 00 00 .y...y...y...y..
00000110 00 FF 00 00 FF 00 00 FF 00 00 FF 00 00 .y...y...y...y..
00000120 00 FF 00 00 FF 00 00 FF 00 00 FF 00 00 .y...y...y...y..
00000130 00 FF 00 00 FF 00 00 FF 00 00 FF 00 00 .y...y...y...y..
00000140 00 FF 00 00 FF 00 00 FF 00 00 FF 00 00 .y...y...y...y..
00000150 00 FF 00 00 FF 00 00 FF 00 00 FF 00 00 .y...y...y...y..
00000160 00 FF 00 00 FF 00 00 FF 00 00 FF 00 00 .y...y...y...y..
00000170 00 FF 00 00 FF 00 00 FF 00 00 FF 00 00 .y...y...y...y..
00000180 00 FF 00 00 FF 00 00 FF 00 00 FF 00 00 .y...y...y...y..
00000190 00 FF 00 00 FF 00 00 FF 00 00 FF 00 00 .y...y...y...y..
000001A0 00 FF 00 00 FF 00 00 FF 00 00 FF 00 00 .y...y...y...y..
000001B0 00 FF 00 00 FF 00 00 FF 00 00 FF 00 00 .y...y...y...y..
000001C0 .. FF .. .. FF .. .. FF .. .. FF .. .. n .. n .. n .. n

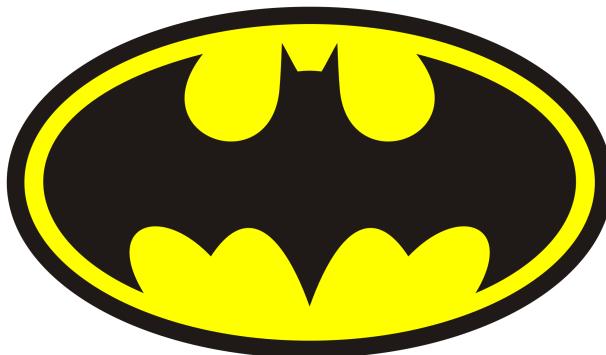
```

(b) Original Hex Data

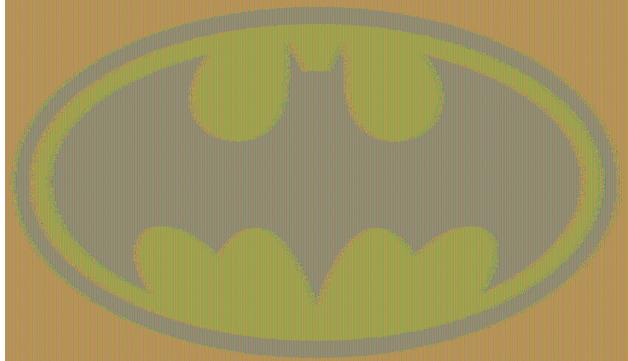
Figure 1: Encryption of Bitmap in Speck 128-128 ECB Mode

1.2.2 Weakness of ECB Mode in Bitmap

Upon encrypting the image and replacing the first encrypted 54 bytes with the BMP File Header and DIB Info Header, the weakness of ECB Mode can be seen in Figure 2(b) as there is original image data being visible after the encryption.



(a) Before Encryption



(b) After Encryption

Figure 2: Encryption of Bitmap in Speck 128-128 ECB Mode

1.3 Encryption of Text File

Using the text shown in Appendix, the text file `test.txt` was encrypted in ECB mode and the 18th byte was then replaced with `0A`. The modified encrypted text file was then decrypted to `test_ecb_modified.txt`. Under the ECB Mode, since every ciphertext is decrypted to its plaintext without any block cipher chaining, hence only the respective plaintext would be

affected by the 18th byte. In this case, the 2nd message block cannot be decrypted back correctly due to the 18th byte being passed through the Speck 128-128 which affected only the rest of the bytes in that message block. Thus, it can be seen in Figure 3(b) that the hex for the 2nd row is still scrambled.

Offset(h)	test.txt	test_ecb_modified.txt
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000000 46 61 72 20 66 61 72 20 61 77 61 79 2C 20 62 65	46 61 72 20 66 61 72 20 61 77 61 79 2C 20 62 65	46 61 72 20 66 61 72 20 61 77 61 79 2C 20 62 65
00000010 68 69 6E 64 20 74 68 65 20 77 6F 72 64 20 6D 6F	68 69 6E 64 20 74 68 65 20 77 6F 72 64 20 6D 6F	68 69 6E 64 20 74 68 65 20 77 6F 72 64 20 6D 6F
00000020 75 6E 74 61 69 6E 73 2C 20 66 61 72 20 66 72 6F	75 6E 74 61 69 6E 73 2C 20 66 61 72 20 66 72 6F	75 6E 74 61 69 6E 73 2C 20 66 61 72 20 66 72 6F
00000030 6D 20 74 68 65 20 63 6F 75 6E 74 72 69 65 73 20	m the countries	m the countries
00000040 56 6F 6B 61 6C 69 61 20 61 6E 64 20 43 6F 6E 73	Vokalia and Cons	Vokalia and Cons
00000050 6F 6E 61 6E 74 69 61 2C 20 74 68 65 72 65 20 6C	onantia, there l	onantia, there l
00000060 69 76 65 20 74 68 65 20 62 6C 69 6E 64 20 74 65	ive the blind te	ive the blind te
00000070 78 74 73 2E 20 53 65 70 61 72 61 74 65 20 74	xts. Separated t	xts. Separated t
00000080 68 65 79 20 60 69 76 65 20 69 69 20 42 6F 6F 6B	hey live in Book	hey live in Book
00000090 6D 61 72 6B 73 67 72 6F 76 65 20 72 69 67 68 74	marksgrave right	marksgrave right
000000A0 20 61 74 20 74 68 65 20 63 6F 61 73 74 20 6F 66	at the coast of	at the coast of
000000B0 20 74 68 65 20 53 65 6D 61 6E 74 69 63 73 2C 20	The Semantics,	The Semantics,
000000C0 61 20 6C 61 72 67 65 20 6C 61 6E 67 75 61 67 65	a large language	a large language
000000D0 20 6F 63 65 61 6E 2E 20 41 20 73 6D 61 6C 20	ocean. A small	ocean. A small
000000E0 72 69 76 65 72 20 6E 61 6D 65 64 20 44 75 64 65	river named Dude	river named Dude
000000F0 62 20 66 6C 6F 77 73 20 62 79 20 74 68 65 69 72	n flows by their	n flows by their
00000100 20 70 6C 61 63 65 20 61 6E 64 20 73 75 70 70 6C	place and suppl	place and suppl
00000110 69 65 73 20 69 74 20 77 69 74 68 20 74 68 65 20	ies it with the	ies it with the
00000120 6E 65 63 73 73 61 72 79 20 72 65 67 65 6C 69	necessary regeli	necessary regeli
00000130 61 6C 69 61 2E 20 49 74 20 69 73 20 61 20 70 61	alia. It is a pa	alia. It is a pa
00000140 72 61 64 69 73 65 6D 61 74 69 63 20 63 6F 75 6E	radisematic coun	radisematic coun
00000150 74 72 79 2C 20 69 6E 20 77 68 69 63 68 20 72 6F	try, in which ro	try, in which ro
00000160 61 73 74 65 64 20 70 61 72 74 73 20 67 66 20 73	asted parts of s	asted parts of s
00000170 65 6E 74 65 6E 63 65 73 20 66 60 79 20 69 6E 74	entences fly int	entences fly int
00000180 6F 20 79 6F 75 72 20 6D 6F 74 68 2E 20 45 76	o your mouth. Ev	o your mouth. Ev
00000190 65 6E 20 74 68 65 20 61 6C 6C 2D 70 6F 77 65 72	en the all-power	en the all-power
000001A0 66 75 6C 20 50 6F 69 6E 74 69 6E 67 20 68 61 73	ful Pointing has	ful Pointing has
000001B0 20 6E 6F 20 63 6F 6E 74 72 6F 6C 20 61 62 6F 75	no control abou	no control abou
~~~~~	~~~~~	~~~~~

(a) Original Text

(b) Decrypted Text after modifying 18th byte

Figure 3: Decryption of Modified Text File using Speck 128-128 ECB Mode

## 2 CBC Mode

### 2.1 Implementation

#### 2.1.1 Initialisation Vector

The CBC Mode is implemented by setting up the Initialisation Vector (IV) using a cryptographically secure random number generator. i.e. /dev/random. Note that the Cygwin Environment was used. Hence, the /dev/random in Cygwin is similar to the CryptGenRandom under the Windows API. Note that in this assignment, the Initialisation Vector was then written into the encrypted file as the first message block. The Code for generating one `unsigned long long` is shown below.

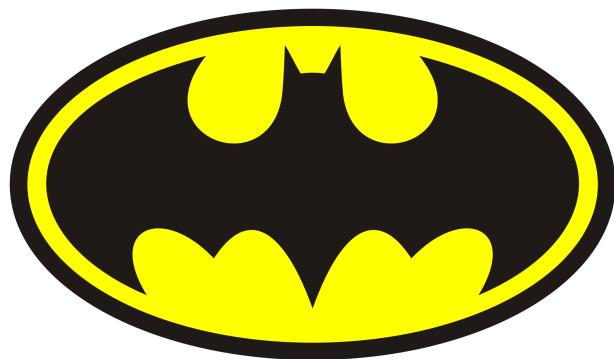
```
unsigned long long InitialVec()
{
    int myFile = open("/dev/random", O_RDONLY);
    unsigned long long rand;
    unsigned long long randomNum = read(myFile, &rand, sizeof(rand)) ;
    close(myFile);
    return rand;
}
```

### 2.1.2 CBC Encryption and Decryption

The encryption in CBC mode goes as follows: The initialisation vector first XORs with the plaintext before passing through the Speck 128-128. The encrypted output which is the ciphertext will also XOR with the next plaintext. Since the encryption takes place after the XOR, padding is required for the plaintext with regards to the last message block. The same padding method from the ECB mode is also applied here. The decryption is also similar. The ciphertext, being also the IV for the next ciphertext, passes through the Speck 128-128 Decryption and XORs with the IV to retrieve back its own plaintext.

## 2.2 Encryption of Bitmap File

Similarly, we placed back the Bitmap File Format into the encrypted hex data. Note that the first message block contains the IV. Hence, we need to remove the IV first before replacing back the 54 bytes. Compared to the ECB Mode, block cipher chaining actually is stronger as now we cannot see any part of the original image visible in the encrypted Bitmap.



(a) Before Encryption



(b) After Encryption

Figure 4: Encryption of Bitmap in Speck 128-128 CBC Mode

## 2.3 Encryption of Text File

Using the same text as the one in Section 1.3, the text file test.txt was encrypted in CBC mode and the 18th byte was then replaced with 0A. The modified encrypted text file was then decrypted to test_cbc_modified.txt. Under the CBC Mode, since there is block cipher chaining, hence the 18th byte which belongs to the 1st message block would affect the 1st and 2nd message block of the decrypted plaintext. In this case, the 1st message block cannot be decrypted back correctly due to the 18th byte being passed through the Speck 128-128 Decryption which affected the rest of the bytes in that message block. Other than the 1st message block, the 18th byte of the 2nd message block of the plaintext is affected and not the rest of it as this is only due to the XOR operation. Thus, it can be seen in Figure 5(b) that the hex for the 1st row and the 18th byte is different.

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00000000	46	61	72	20	66	61	72	20	61	77	61	79	2C	20	62	65
00000010	68	69	6E	64	20	74	68	65	20	77	6F	72	64	20	6D	6F
00000020	75	6E	74	61	65	6E	73	2C	20	66	61	72	20	66	72	6F
00000030	6D	20	74	68	65	20	63	67	75	6E	74	72	69	65	73	20
00000040	56	6F	6B	61	6C	69	61	20	61	6E	64	20	43	6F	6E	73
00000050	6F	6E	61	6E	74	69	61	2C	20	74	68	65	72	65	20	6C
00000060	69	76	65	20	74	68	65	20	62	6C	69	6E	64	20	74	65
00000070	78	74	73	2E	20	53	65	70	61	72	61	74	65	20	74	74
00000080	68	65	79	20	6C	69	76	65	20	69	6E	20	42	6F	6B	6B
00000090	6D	61	72	6B	73	67	72	6F	76	65	20	72	69	67	68	74
000000A0	20	61	74	70	68	65	20	63	6F	61	73	74	20	6F	66	66
000000B0	20	74	68	65	20	53	65	6D	61	6E	74	69	63	73	2C	20
000000C0	61	20	6C	61	72	67	65	20	6C	61	6E	67	75	61	67	65
000000D0	20	6F	63	65	61	6E	2E	20	41	20	73	6D	61	6C	6C	20
000000E0	72	69	76	65	72	20	6E	61	6D	65	64	20	44	75	64	65
000000F0	6E	20	66	6C	6F	77	73	20	62	79	20	74	68	65	69	72
00000100	20	70	6C	61	63	65	20	61	6E	64	20	73	75	70	6C	6C
00000110	69	65	73	20	69	74	20	77	69	74	60	20	74	68	65	20
00000120	6E	65	63	65	73	73	61	72	79	20	72	65	67	65	6C	69
00000130	61	6C	69	61	2E	20	49	74	20	69	73	20	61	20	70	61
00000140	72	61	64	69	73	65	6D	61	74	69	63	20	63	6F	75	6E
00000150	74	72	79	2C	20	69	6E	20	77	68	69	68	20	72	6F	6F
00000160	61	73	74	65	64	20	70	61	72	74	73	20	6E	66	20	73
00000170	65	6E	74	65	6E	63	65	73	20	66	6C	79	20	69	6E	74
00000180	6F	20	79	6F	75	72	20	6D	6F	75	74	68	20	45	76	76
00000190	65	6E	20	74	68	65	20	61	6C	6C	2D	70	67	77	65	72
000001A0	66	75	6C	20	50	6F	69	74	69	6E	20	68	61	73	ful	Pointing has
000001B0	20	6E	6F	20	63	6F	6E	74	72	6F	6C	20	61	62	6F	75
000001C0	74	20	74	69	6E	20	67	68	61	73	20	66	61	73	+ the blind te	nt

(a) Original Text

(b) Decrypted Text after modifying 18th byte

Figure 5: Decryption of Modified Text File using Speck 128-128 CBC Mode

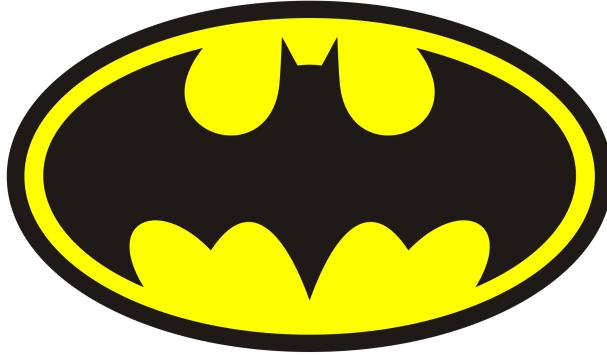
### 3 OFB Mode

#### 3.1 Implementation

The OFB also uses the same method of generating IV as mentioned in Section 2.1.1. In OFB Mode, since the Speck 128-128 Encryption is used for the IV and not the plaintext, hence there is no padding required as we could just XOR the plaintext with the encrypted IV to obtain the ciphertext.

## 3.2 Encryption of Bitmap File

Similar to CBC Mode, the original image data is not visible in the encrypted output. However, we note the slight differences in RGB pixels as compared to the one in CBC mode.



(a) Before Encryption



(b) After Encryption

Figure 6: Encryption of Bitmap in Speck 128-128 OFB Mode

## 3.3 Encryption of Text File

Using the same text as the one in Section 1.3, the text file test.txt was encrypted in OFB mode and the 18th byte was then replaced with 0A. The modified encrypted text file was then decrypted to test_ofb_modified.txt. Under the OFB Mode, only the 18th byte will be decrypted wrongly since the ciphertext does not pass through the Block Cipher and only XORs with the encrypted IV. Hence, we can see in Figure 7 that the hex data only differ by the 18th byte.

	test.txt	test_ecb_modified.txt	test_ofb_modified.txt
Offset(h)	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000000	46 61 72 20 66 61 72 20 61 77 61 79 2C 20 62 65	Far far away, be	Far far away, be
00000010	68 69 6E 64 74 68 65 20 77 6F 72 64 20 6D 6F	hind the word mo	hind the word mo
00000020	75 6E 74 61 69 6E 73 2C 20 66 61 72 20 66 72 6F	untains, far fro	untains, far fro
00000030	6D 20 74 68 65 20 63 6F 75 6E 74 72 69 65 73 20	m the countries	m the countries
00000040	56 6F 6B 61 69 61 20 61 6E 64 20 43 6F 6E 73	Vokalia and Cons	Vokalia and Cons
00000050	6F 6E 61 6E 74 69 61 2C 20 74 68 65 73 20 6C	onantia, there l	onantia, there l
00000060	69 76 65 20 74 68 65 20 62 6C 6E 64 20 74 65	ive the blind te	ive the blind te
00000070	78 74 73 2E 20 53 65 70 61 72 74 65 64 20 74	xts. Separated t	xts. Separated t
00000080	68 65 79 20 6C 69 76 65 20 69 6E 20 42 6F 6F 6B	hey live in Book	hey live in Book
00000090	6D 61 72 6B 73 67 72 6F 65 20 72 69 67 68 74	marksgrave right	marksgrave right
000000A0	20 61 74 20 74 68 65 20 63 6F 61 73 74 20 6F 66	at the coast of	at the coast of
000000B0	20 74 68 65 20 53 65 6D 61 6E 74 69 63 73 2C 20	the Semantics,	the Semantics,
000000C0	61 6C 61 72 67 65 20 6C 61 6E 75 61 67 65	a large language	a large language
000000D0	20 6F 63 65 61 6E 2E 20 41 20 73 6D 61 6C 6C 20	ocean. A small	ocean. A small
000000E0	72 69 76 65 72 20 6E 61 6D 65 20 44 75 64 65	river named Dude	river named Dude
000000F0	6E 20 66 6C 6F 77 73 20 62 79 20 74 68 65 69 72	n flows by their	n flows by their
00000100	20 70 6C 61 63 65 20 61 6E 64 20 73 75 70 70 6C	place and suppl	place and suppl
00000110	69 65 73 20 69 74 20 77 69 74 68 20 74 68 65 20	ies it with the	ies it with the
00000120	6E 65 63 65 73 73 61 72 79 20 72 65 67 65 6C 69	necessary regeli	necessary regeli
00000130	61 6C 69 61 2E 20 49 74 20 69 73 20 61 20 70 61	alia. It is a pa	alia. It is a pa
00000140	72 61 64 69 73 65 6D 61 74 69 63 20 63 6F 75 6E	radisematic coun	radisematic coun
00000150	74 72 79 2C 20 69 6E 20 77 68 69 63 68 20 72 6F	try, in which ro	try, in which ro
00000160	61 73 74 65 64 20 70 61 72 74 73 20 68 66 20 73	asted parts of s	asted parts of s
00000170	65 6E 74 65 6E 63 65 73 20 66 6C 79 20 69 6E 74	entences fly int	entences fly int
00000180	6F 20 79 6F 75 72 20 6D 6F 75 74 68 2E 20 45 76	o your mouth. Ev	o your mouth. Ev
00000190	65 6E 20 74 68 65 20 61 6C 2D 70 6F 77 65 72	en the all-power	en the all-power
000001A0	66 75 6C 20 50 6F 69 6E 74 69 6E 67 20 68 61 73	ful Pointing has	ful Pointing has
000001B0	20 6E 6F 20 63 6F 6E 74 72 6F 6C 20 61 62 6F 75	no control abou	no control abou

(a) Original Text

(b) Decrypted Text after modifying 18th byte

Figure 7: Decryption of Modified Text File using Speck 128-128 OFB Mode