CHAPTER 5
IMPLEMENTATION AND TESTING

5.1 Implementation

This section will explain the utilization of the code that is used in this study.

5.1.1 Image Input

The following is the code for inputting an image:

```
1. List<Asset> resultList = await MultiImagePicker.pickImages(
2.       maxImages: 1,
3. );
4. ByteData byteData = await resultList[0].getByteData(
5.       quality: 75
6. );
7. final buffer = byteData.buffer;
8. var list = buffer.asUint8List(byteData.offsetInBytes,
   byteData.lengthInBytes);
9. base64Image = base64Encode(list);
```

Lines 1-3 are the commands used to take pictures from a smartphone. Lines 4-6 are the commands used to extract assets from the selected image. Line 7 is to retrieve the object from the buffer byte. Line 8 is functioned to change the byte buffer to Uint8List. On the line 9, the results of line 8 are then changed back to base64.

5.1.2 ECB Encryption Mode

The results of the implementation of the ECB DES mode encryption algorithm can be seen as below.

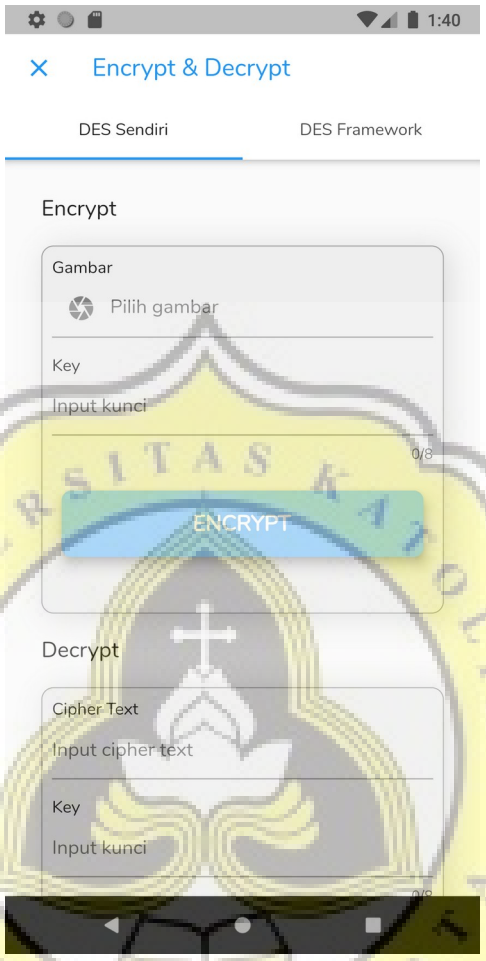

Figure 5.1: ECB Mode Encryption Display

To encrypt, select the image first. To select an image, press the "Select image" button, then the page will show the gallery found on an Android smartphone. After selecting the image, the next step is to enter the key. The key has a length of 8 digits. After selecting an image and entering the key, then press the "ENCRYPT" button to execute the program.

These are the code for doing encryption:

```
1. var hasilSplit = _allFunction.splitTo64Bit(kata[h]);
2. var binaryKey = _allFunction.stringToBinary(key);
3. var resultPermutationKey =
   _allFunction.permutationWithTablePC1(binaryKey);
4. for (var j = 0; j < shift.length; j++) {
```

```
5. var c = _allFunction.leftShifting(cidi[j].ci, shift[j]);
6. var d = _allFunction.leftShifting(cidi[j].di, shift[j]);
7. cidi.add(CiDi(c, d));
8.}
9.var k = _allFunction.permutationWithTablePC2(joinCiDi);
10.for (var i = 0; i < hasilSplit.length; i++) {
11.  var binary = _allFunction.stringToBinary(hasilSplit[i]);
12.  var resultPermutationPlainText =
  _allFunction.permutationWithTableIP(binary);
13.  var er0 = _allFunction.expansiWithTableExpansi(r0);
14.  var a1 = _allFunction.xOr(er0, ki[0].k);
15.  var b1 = _allFunction.subtitusiSBox(a1);
16.  var pb1 = _allFunction.permutationWithTablePBox(b1);
17.  var r1 = _allFunction.xOr(pb1, l0);
18.  var er1 = _allFunction.expansiWithTableExpansi(r1);
19.  var resultAll =
  _allFunction.permutationWithTableIP1(joinR16L16);
20.  var _resultSplit = _allFunction.splitTo4Bit(resultAll);
21.  temp = _allFunction.binaryToHex(_resultSplit);
22.}
```

Line 1 contains the code to change the plaintext into 8 bytes or 64 bits which will be saved in the hasilSpit variable in the form of an array. Lines 2 and 3 are functioned to change the string to binary from the key which is then mutated with PC1 table. Lines 4-8 are 16 times key-shift command which will be stored in the cidi array of objects. Line 9 is the result of cidi permutation with PC2 table. After that, lines 10-22 are encryption for every 8 bytes. Line 11 is functioned to change the plaintext into binary that will be mutated with the IP table in line 12 and be expanded with the Expansion table in line 13. Furthermore, there will be 16x interactions in line 14-18, then in line 19 the interaction results will be mutated with IP1 table. Lastly, the binary is changed into hexadecimal in line 21.

5.1.3 ECB Decryption Mode

The results of the implementation of the DES ECB mode decryption algorithm can be seen as below.

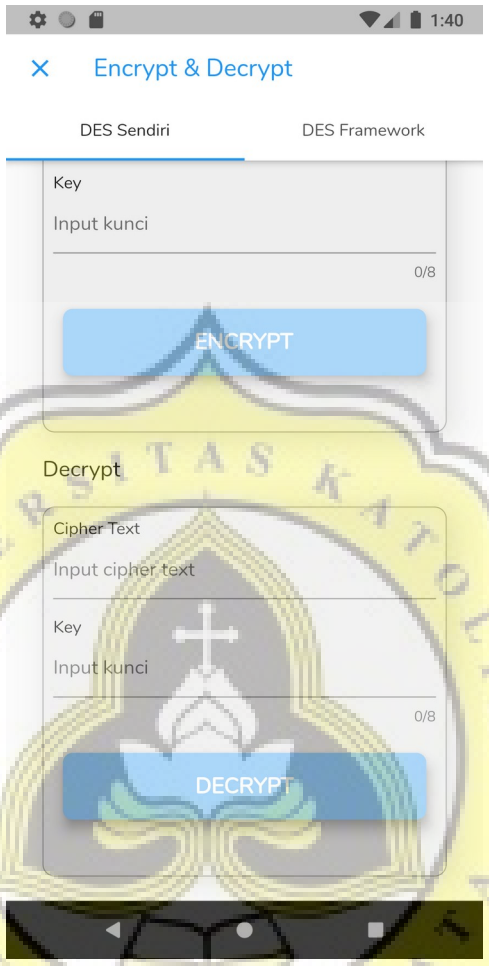

Figure 5.2: ECB Mode Decryption Display

To decrypt, the results of the image encryption will be filled in the ciphertext input selection section. After that, enter the same key when doing encryption that has a length of 8 digits. Then, press the "DECRYPT" button and the results of the decryption will be displayed at the bottom of the button.

These are the code for doing decryption:

```
1. var binaryKey = _function.stringToBinary(key);
2. var resultPermutationKey =
   _function.permutationWithTablePC1(binaryKey);
3. for (var j = 0; j < shift.length; j++) {
4.   var c = _function.leftShifting(cidi[j].ci, shift[j]);
5.   var d = _function.leftShifting(cidi[j].di, shift[j]);
6.   cidi.add(CiDi(c, d));
7. }
8. var k = _function.permutationWithTablePC2(joinCiDi);
9. var hasilSplit =
   _allFunction.splitTo128Bit(splitCipherText[h]);
10.  for (var i = 0; i < hasilSplit.length; i++) {
11.     var binary = _allFunction.hexToBinary(hasilSplit[i]);
12.     var resultPermutationPlainText =
   _allFunction.permutationWithTableIP(binary);
13.     var er0 = _allFunction.expansiWithTableExpansi(r0);
14.     var a1 = _allFunction.xOr(er0, ki[15].k);
15.     var b1 = _allFunction.subtitusiSBox(a1);
16.     var pb1 = _allFunction.permutationWithTablePBox(b1);
17.     var r1 = _allFunction.xOr(pb1, l0);
18.     var er1 = _allFunction.expansiWithTableExpansi(r1);
19.     var resultAll =
   _allFunction.permutationWithTableIP1(joinR16L16);
20.     temp = _allFunction.binaryToString(resultAll);
21.  }
```

Lines 1 and 2 are functioned to change the string to binary from the key which is then mutated with PC1 table. Lines 3-7 represent key shift orders at 16 times which will be stored in the cidi objects array. Row 8 is the result of cidi permutation with PC2 table. At line 9, the ciphertext is converted into several parts whose returns is array. After that, lines 10-21 are decrypted for each of the 16 digits. Line 11 is a program to change hexadecimal to binary, which will be mutated with IP table in line 12. Furthermore, there are 16x interactions in lines 13-18 and are mutated with IP1 table in line 19. Lastly, the binary is changed back to a plaintext in line 20.

5.1.4 CBC Encryption Mode

The results of implementing the CBC DES mode encryption algorithm can be seen as below.

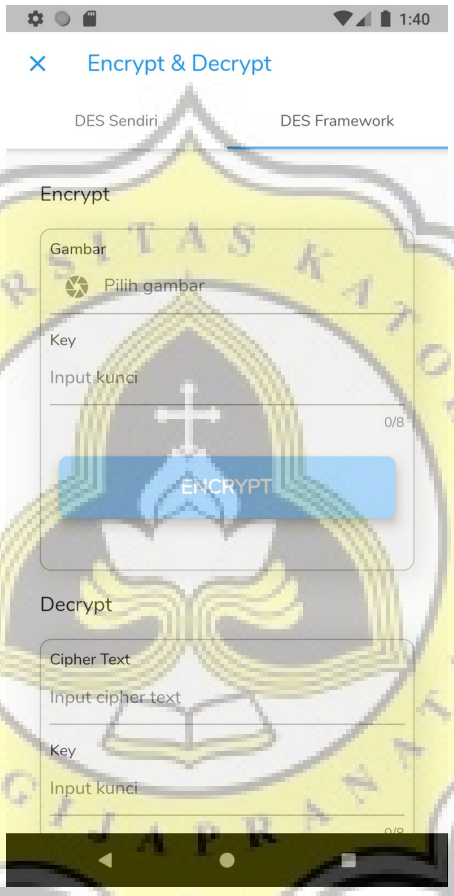

Figure 5.3: CBC Mode Encryption Display

To encrypt, select the image first. To select an image press the "Select image" button, then the page will open the gallery from Android smartphone. After selecting the image, the next step is to enter the key which has a length of 8 digits. After selecting an image and entering the key, then press the "ENCRYPT" button to execute the program.

These are the code for doing encryption:

```
1.import 'package:flutter_des/flutter_des.dart';
2.onPressed: () async {
3.  print(DateTime.now());
4.   var enkrip = await FlutterDes.encryptToHex(_imageLampiranWA,
  _keyController.text, iv: "12345678");
5.  print(DateTime.now());
6.   print("PANJANG = ${enkrip.length}");
7.},
```

Line 1 is functioned to import the library that will be used in line 4. Lines 2 and 7 are the lines for special functions in it. Lines 3 and 5 are functioned to display the time that will later be used for doing calculations. Line 4 is the command for encryption of the base64 image with a predefined key. This library will automatically adjust the length of the plaintext itself, so the length of the plaintext doesn't need to be worried.

5.1.5 CBC Decryption Mode

The results of the implementation of DES mode CBC decryption algorithm can be seen as below:

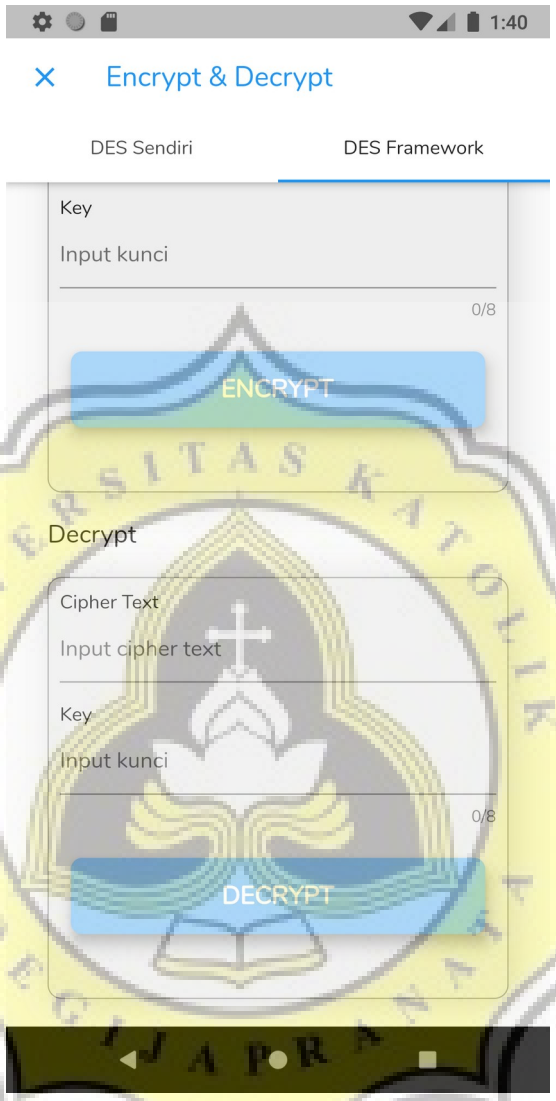

Figure 5.4: CBC Mode Decryption Display

To decrypt, the results of the image encryption will be filled in the ciphertext input selection section. After that, enter the same key as before when doing encryption which has a length of 8 digits. Then, press the "DECRYPT" button and finally the results of the decryption will be displayed at the bottom of the button.

These are the code for doing decryption:

```
1. import 'package:flutter_des/flutter_des.dart';
2. onPressed: () async {
3.           print(DateTime.now());
4.           var dekrip = await
   FlutterDes.decryptFromHex(_cipherTextController.text,
   _keyController.text, iv: "12345678");
5.           print(DateTime.now());
6.           print("PANJANG = ${dekrip.length}");
7. },
```

Line 1 is functioned to import the library that will be used in line 4. Lines 2 and 7 are the lines for special functions within it. Lines 3 and 5 are functioned to display the time that will be used for doing calculations later. Line 4 is the command to decrypt the ciphertext with the same key used when encrypting. This library will adjust the length of the ciphertext itself so the length of the plaintext doesn't need to be worried.

5.2 Testing

5.2.1 Encryption

These ten pictures are going to be tested. These pictures are:

1. Mario.png (240x320)

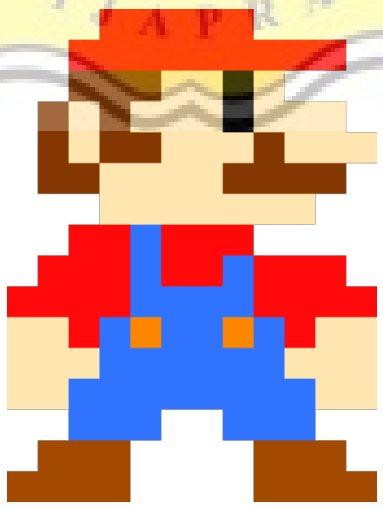

Figure 5.5: Image Test 1

2. Rabbit.jpg (336x339)

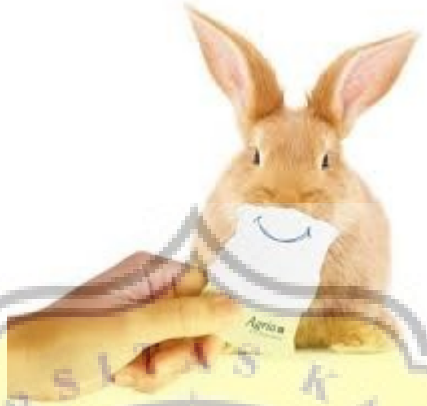

Figure 5.6: Image Test 2

3. Shoes.png (480x359)

Figure 5.7: Image Test 3

4. Headset.jpg (385x385)

Figure 5.8: Image Test 4

5. Tiger.jpg (387x791)

Figure 5.9: Image Test 5

6. Vector.jpg (626x417)

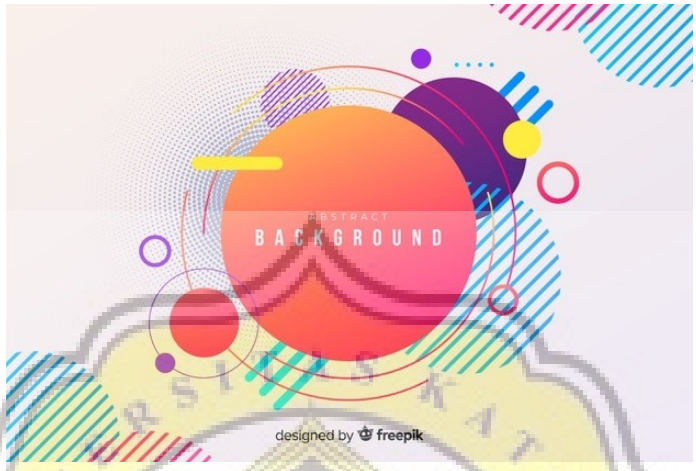

Figure 5.10: Image Test 6

7. Smartphone.jpg (350x230)

Figure 5.11: Image Test 7

8. Smartphone.jpeg (600x400)

Figure 5.12: Image Test 8

9. Flashdisk.jpg (380x380)

Figure 5.13: Image Test 9

10. Bottle.jpg (348x363)

Figure 5.14: Image Test 10

Here is the encryption time table in milliseconds:

Table 5.1: Table Encryption Time (in millisecond)

	Plaintext Length	ECB Mode		CBC Mode	
		Time	Ciphertext Length	Time	Ciphertext Length
Mario.png	6180	460,344	12368	58,544	12368
Rabbit.jpg	12856	952,272	25712	10,904	25728
Shoes.png	31996	2380,040	64000	16,520	64000
Headset.jpg	22428	1841,737	44864	18,573	44864
Tiger.jpg	108212	8911,640	216432	52,259	216432
Vector.jpg	43464	3288,386	86928	21,970	86944
Smartphone.jpg	13616	1109,799	27232	10,842	27248
Smartphone.jpeg	35888	2761,246	71776	31,215	71792
Flashdisk.jpg	12076	904,024	24160	12,989	24160
Bottle.jpg	7128	591,607	14256	9,155	14272

The results of testing the image encryption process are in the table above. The time required for encryption with the ECB DES Mode Algorithm is longer than that of the CBC DES Mode Algorithm. This is because in ECB Mode the

process of executing the code that is run sequentially based on the code written and the received plaintext will be checked in length first. If the length of the plaintext is 16 bytes, the plaintext will be divided into two part (8 bytes each) which are then encrypted one by one.

While the CBC DES Mode Algorithm has a relatively shorter time because the code execution process is asynchronous. The process of executing this code is not in accordance with the existing order so that if the length of the plaintext is 16 bytes, the encryption process can run simultaneously and the results of the encryption are combined again. The total time difference from the average encryption results between ECB and CBC Modes is 2296,812 milliseconds.

5.2.2 Decryption

Here is the decryption time table in milliseconds:

Table 5.2: Table Decryption Time (in millisecond)

	ECB Mode			CBC Mode		
	Ciphertext Length	Time	Plaintext Length	Ciphertext Length	Time	Plaintext Length
Mario.png	12368	464,933	6180	12368	36,553	6180
Rabbit.jpg	25712	960,663	12856	25728	60,781	12856
Shoes.png	64000	2432,750	31996	64000	32,075	31996
Headset.jpg	44864	1762,887	22428	44864	19,966	22428
Tiger.jpg	216432	9500,259	108212	216432	55,396	108212
Vector.jpg	86928	3714,853	43464	86944	20,604	43464
Smartphone.jpg	27232	1021,303	13616	27248	10,761	13616
Smartphone.jpeg	71776	2839,548	35888	71792	22,972	35888
Flashdisk.jpg	24160	892,069	12076	24160	11,739	12076
Bottle.jpg	14256	486,244	7128	14272	41,987	7128

Table 5.2 is a result table for testing the decryption process. The time required for decryption using the ECB DES mode algorithm is longer than that of the CBC DES mode algorithm. This is the same as, during encryption, the code

execution process that is run on the ECB DES mode algorithm is sequentially based on the code written. So if the length of the ciphertext is 32 digits, then the ciphertext will be divided into two parts first (every 16 digits) which is then decrypted one by one and then put together into a plaintext.

CBC DES mode algorithm has a relatively shorter time because the code execution process is asynchronous. This makes the decryption process using the CBC DES mode algorithm have a shorter time. The total time difference from the average decryption result between ECB and CBC modes is 2376,268 milliseconds.