APPENDIX

Main.dart

```
import 'package:flutter/material.dart';
import 'package:salmon/ui/des.dart';
void main() {
  WidgetsFlutterBinding.ensureInitialized();
  SharedPreferences.getInstance().then((prefs) {
    runApp(MyApp(prefs: prefs));
  });
}
class MyApp extends StatelessWidget {

  final SharedPreferences prefs;
  MyApp({this.prefs});
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: "SAMPERIN",
      theme: ThemeData(
        primaryColor: Colors.blue[200],
        fontFamily: 'Nunito-Regular',
      ),
      home: DesBener(),
      debugShowCheckedModeBanner: false,
    );
  }
}
```

Des.dart

```
import 'package:flutter/material.dart';
import 'package:salmon/ui/tabBar1.dart';
import 'package:salmon/ui/tabBar2.dart';

class DesBener extends StatefulWidget {
  @override
  _DesBenerState createState() => _DesBenerState();
}

class _DesBenerState extends State<DesBener> with
SingleTickerProviderStateMixin {

  TabController tabbarController;

  @override
  void initState() {
    tabbarController = TabController(vsync: this, length: 2);
    super.initState();
  }
```

```
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        backgroundColor: Colors.white,
        iconTheme: IconThemeData(color: Colors.blue),
        leading: IconButton(
          icon: Icon(Icons.close),
          color: Colors.blue,
          onPressed: () {
            Navigator.of(context).pop();
          },
        ),
        title: Text(
          "Encrypt & Decrypt",
          style: TextStyle(
            color: Colors.blue,
          ),
        ),
        bottom: TabBar(
          controller: tabbarController,
          tabs: <Widget>[
            Tab(
              child: Text("DES Sendiri"),
            ),
            Tab(
              child: Text("DES Framework"),
            ),
          ],
        ),
      ),
      body: TabBarView(
        controller: tabbarController,
        children: <Widget>[
          TabBar1(),
          TabBar2()
        ],
      ),
    );
  }
}
```

TabBar1.dart

```
import 'dart:convert';
import 'dart:typed_data';
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';
import 'package:salmon/ui/allFunction.dart';
import 'package:multi_image_picker/multi_image_picker.dart';

class TabBar1 extends StatefulWidget {
  @override
  _TabBar1State createState() => _TabBar1State();
```

```
}

class _TabBar1State extends State<TabBar1> {

  var _cipherTextController = TextEditingController();
  var _keyController = TextEditingController();
  Uint8List _fotoLampiranWA;
  Uint8List _hasilJadi;
  var _imageLampiranWA;

  var _resultEncryptHex = "";
  var _resultDecrypt = "";

  @override
  void initState() {
    super.initState();
  }

  Future _openImages() async {
    String base64Image;
    try {
      List<Asset> resultList = await MultiImagePicker.pickImages(
        maxImages: 1,
      );
      ByteData byteData = await resultList[0].getByteData(quality:
75);
      final buffer = byteData.buffer;
      var list = buffer.asUint8List(byteData.offsetInBytes,
byteData.lengthInBytes);
      base64Image = base64Encode(list);
      setState(() {
        _imageLampiranWA = base64Image;
        _fotoLampiranWA = base64Decode(base64Image);
      });
      print("BASE 64 = ${_imageLampiranWA.length}");
    }
    catch (e) {
      print(e);
    }
  }

// ======================================================
// ||====================================================||
// ||                                                    ||
// ||                      ENCRYPT                       ||
// ||                                                    ||
// ||====================================================||
// ======================================================

  String _encrypt(String plainText, String key) {

    var _function = AllFunction();
    var _hasilEncrypt = "";
```

```
    var kata = _function.splitTo1024Char(plainText);

    for (var h = 0; h < kata.length; h++) {
      var _allFunction = AllFunction();
      var cidi = List<CiDi>();
      var ki = List<K>();

      var temp;

      // Hasil split menjadi masing-masing 64 bit
      var hasilSplit = _allFunction.splitTo64Bit(kata[h]);

      // Mengubah string ke binary
      var binaryKey = _allFunction.stringToBinary(key);

      // Permutasi key dengan tabel PC1
      var resultPermutationKey =
_allFunction.permutationWithTablePC1(binaryKey);

      // Membagi hasil permutasi key menjadi 2 bagian c0 dan d0
      var c0 = List<String>();
      var d0 = List<String>();
      for (var j = 0; j < 28; j++) {
        c0.add(resultPermutationKey[j]);
      }
      for (var j = 28; j < 56; j++) {
        d0.add(resultPermutationKey[j]);
      }

      // Ditampung dengan array of object
      cidi.add(CiDi(c0, d0));

      // Perputaran sebanyak 16x
      var shift = List<int>();
      shift = [
        1, 1, 2, 2, 2, 2, 2, 2,
        1, 2, 2, 2, 2, 2, 2, 1
      ];
      for (var j = 0; j < shift.length; j++) {
        var c = _allFunction.leftShifting(cidi[j].ci, shift[j]);
        var d = _allFunction.leftShifting(cidi[j].di, shift[j]);
        cidi.add(CiDi(c, d));
      }

      // Permutasi hasil perputaran dengan tabel PC2
      for (var j = 1; j < cidi.length; j++) {
        var joinCiDi = List<String>();
        for (var k = 0; k < cidi[j].ci.length; k++) {
          joinCiDi.add(cidi[j].ci[k]);
        }
        for (var k = 0; k < cidi[j].di.length; k++) {
          joinCiDi.add(cidi[j].di[k]);
        }
        var k = _allFunction.permutationWithTablePC2(joinCiDi);
```

```
        ki.add(K(k));
      }

      // Proses plaintext
      for (var i = 0; i < hasilSplit.length; i++) {

        // Mengubah string ke binary
        var binary = _allFunction.stringToBinary(hasilSplit[i]);
        // Hasil permutasi dengan tabel IP
        var resultPermutationPlainText =
_allFunction.permutationWithTableIP(binary);

        // Membagi hasil permutasi plaintext menjadi 2 bagian l0
dan r0
        var l0 = List<String>();
        var r0 = List<String>();
        for (var j = 0; j < 32; j++) {
          l0.add(resultPermutationPlainText[j]);
        }
        for (var j = 32; j < 64; j++) {
          r0.add(resultPermutationPlainText[j]);
        }

        // Interaksi ke-1
        var er0 = _allFunction.expansiWithTableExpansi(r0);
        var a1 = _allFunction.xOr(er0, ki[0].k);
        var b1 = _allFunction.subtitusiSBox(a1);
        var pb1 = _allFunction.permutationWithTablePBox(b1);
        var r1 = _allFunction.xOr(pb1, l0);
        var er1 = _allFunction.expansiWithTableExpansi(r1);

        // Interaksi ke-2
        var a2 = _allFunction.xOr(er1, ki[1].k);
        var b2 = _allFunction.subtitusiSBox(a2);
        var pb2 = _allFunction.permutationWithTablePBox(b2);
        var r2 = _allFunction.xOr(pb2, r0);
        var er2 = _allFunction.expansiWithTableExpansi(r2);

        // Interaksi ke-3
        var a3 = _allFunction.xOr(er2, ki[2].k);
        var b3 = _allFunction.subtitusiSBox(a3);
        var pb3 = _allFunction.permutationWithTablePBox(b3);
        var r3 = _allFunction.xOr(pb3, r1);
        var er3 = _allFunction.expansiWithTableExpansi(r3);

        // Interaksi ke-4
        var a4 = _allFunction.xOr(er3, ki[3].k);
        var b4 = _allFunction.subtitusiSBox(a4);
        var pb4 = _allFunction.permutationWithTablePBox(b4);
        var r4 = _allFunction.xOr(pb4, r2);
        var er4 = _allFunction.expansiWithTableExpansi(r4);

        // Interaksi ke-5
        var a5 = _allFunction.xOr(er4, ki[4].k);
```

```
var b5 = _allFunction.subtitusiSBox(a5);
var pb5 = _allFunction.permutationWithTablePBox(b5);
var r5 = _allFunction.xOr(pb5, r3);
var er5 = _allFunction.expansiWithTableExpansi(r5);

// Interaksi ke-6
var a6 = _allFunction.xOr(er5, ki[5].k);
var b6 = _allFunction.subtitusiSBox(a6);
var pb6 = _allFunction.permutationWithTablePBox(b6);
var r6 = _allFunction.xOr(pb6, r4);
var er6 = _allFunction.expansiWithTableExpansi(r6);

// Interaksi ke-7
var a7 = _allFunction.xOr(er6, ki[6].k);
var b7 = _allFunction.subtitusiSBox(a7);
var pb7 = _allFunction.permutationWithTablePBox(b7);
var r7 = _allFunction.xOr(pb7, r5);
var er7 = _allFunction.expansiWithTableExpansi(r7);

// Interaksi ke-8
var a8 = _allFunction.xOr(er7, ki[7].k);
var b8 = _allFunction.subtitusiSBox(a8);
var pb8 = _allFunction.permutationWithTablePBox(b8);
var r8 = _allFunction.xOr(pb8, r6);
var er8 = _allFunction.expansiWithTableExpansi(r8);

// Interaksi ke-9
var a9 = _allFunction.xOr(er8, ki[8].k);
var b9 = _allFunction.subtitusiSBox(a9);
var pb9 = _allFunction.permutationWithTablePBox(b9);
var r9 = _allFunction.xOr(pb9, r7);
var er9 = _allFunction.expansiWithTableExpansi(r9);

// Interaksi ke-10
var a10 = _allFunction.xOr(er9, ki[9].k);
var b10 = _allFunction.subtitusiSBox(a10);
var pb10 = _allFunction.permutationWithTablePBox(b10);
var r10 = _allFunction.xOr(pb10, r8);
var er10 = _allFunction.expansiWithTableExpansi(r10);

// Interaksi ke-11
var a11 = _allFunction.xOr(er10, ki[10].k);
var b11 = _allFunction.subtitusiSBox(a11);
var pb11 = _allFunction.permutationWithTablePBox(b11);
var r11 = _allFunction.xOr(pb11, r9);
var er11 = _allFunction.expansiWithTableExpansi(r11);

// Interaksi ke-12
var a12 = _allFunction.xOr(er11, ki[11].k);
var b12 = _allFunction.subtitusiSBox(a12);
var pb12 = _allFunction.permutationWithTablePBox(b12);
var r12 = _allFunction.xOr(pb12, r10);
var er12 = _allFunction.expansiWithTableExpansi(r12);
```

```
        // Interaksi ke-13
        var a13 = _allFunction.xOr(er12, ki[12].k);
        var b13 = _allFunction.subtitusiSBox(a13);
        var pb13 = _allFunction.permutationWithTablePBox(b13);
        var r13 = _allFunction.xOr(pb13, r11);
        var er13 = _allFunction.expansiWithTableExpansi(r13);

        // Interaksi ke-14
        var a14 = _allFunction.xOr(er13, ki[13].k);
        var b14 = _allFunction.subtitusiSBox(a14);
        var pb14 = _allFunction.permutationWithTablePBox(b14);
        var r14 = _allFunction.xOr(pb14, r12);
        var er14 = _allFunction.expansiWithTableExpansi(r14);

        // Interaksi ke-15
        var a15 = _allFunction.xOr(er14, ki[14].k);
        var b15 = _allFunction.subtitusiSBox(a15);
        var pb15 = _allFunction.permutationWithTablePBox(b15);
        var r15 = _allFunction.xOr(pb15, r13);
        var er15 = _allFunction.expansiWithTableExpansi(r15);

        // Interaksi ke-16
        var a16 = _allFunction.xOr(er15, ki[15].k);
        var b16 = _allFunction.subtitusiSBox(a16);
        var pb16 = _allFunction.permutationWithTablePBox(b16);
        var r16 = _allFunction.xOr(pb16, r14);
        var er16 = _allFunction.expansiWithTableExpansi(r16);

        var joinR16L16 = List<String>();
        for (var j = 0; j < r16.length; j++) {
          joinR16L16.add(r16[j]);
        }
        for (var j = 0; j < r15.length; j++) {
          joinR16L16.add(r15[j]);
        }

        var resultAll =
_allFunction.permutationWithTableIP1(joinR16L16);
        var _resultSplit = _allFunction.splitTo4Bit(resultAll);
        temp = _allFunction.binaryToHex(_resultSplit);
      }
      _hasilEncrypt = _hasilEncrypt + temp;
    }
    return _hasilEncrypt;
  }

// ======================================================
// ||====================================================||
// ||                                                    ||
// ||                    DECRYPT                         ||
// ||                                                    ||
// ||====================================================||
// ======================================================
  String _decrypt(String cipherText, String key) {
```

```
    var _function = AllFunction();
    var _hasilDecrypt = "";
    var cidi = List<CiDi>();
    var ki = List<K>();

    // Mengubah string ke binary
    var binaryKey = _function.stringToBinary(key);

    var splitCipherText = _function.splitTo100000Char(cipherText);

    // Permutasi key dengan tabel PC1
    var resultPermutationKey =
_function.permutationWithTablePC1(binaryKey);

    // Membagi hasil permutasi key menjadi 2 bagian c0 dan d0
    var c0 = List<String>();
    var d0 = List<String>();
    for (var j = 0; j < 28; j++) {
      c0.add(resultPermutationKey[j]);
    }
    for (var j = 28; j < 56; j++) {
      d0.add(resultPermutationKey[j]);
    }

    // Ditampung dengan array of object
    cidi.add(CiDi(c0, d0));

    // Perputaran sebanyak 16x
    var shift = List<int>();
    shift = [
      1, 1, 2, 2, 2, 2, 2, 2,
      1, 2, 2, 2, 2, 2, 2, 1
    ];
    for (var j = 0; j < shift.length; j++) {
      var c = _function.leftShifting(cidi[j].ci, shift[j]);
      var d = _function.leftShifting(cidi[j].di, shift[j]);
      cidi.add(CiDi(c, d));
    }

    // Permutasi hasil perputaran dengan tabel PC2
    for (var j = 1; j < cidi.length; j++) {
      var joinCiDi = List<String>();
      for (var k = 0; k < cidi[j].ci.length; k++) {
        joinCiDi.add(cidi[j].ci[k]);
      }
      for (var k = 0; k < cidi[j].di.length; k++) {
        joinCiDi.add(cidi[j].di[k]);
      }
      var k = _function.permutationWithTablePC2(joinCiDi);
      ki.add(K(k));
    }

    for (var h = 0; h < splitCipherText.length; h++) {
```

```
      var _allFunction = AllFunction();
      var temp = "";

      // Hasil split menjadi masing-masing 64 bit
      var hasilSplit =
_allFunction.splitTo128Bit(splitCipherText[h]);

      // Proses plaintext
      for (var i = 0; i < hasilSplit.length; i++) {
        // Mengubah hex ke binary
        var binary = _allFunction.hexToBinary(hasilSplit[i]);
        // Hasil permutasi dengan tabel IP
        var resultPermutationPlainText =
_allFunction.permutationWithTableIP(binary);

        // Membagi hasil permutasi plaintext menjadi 2 bagian l0
dan r0
        var l0 = List<String>();
        var r0 = List<String>();
        for (var j = 0; j < 32; j++) {
          l0.add(resultPermutationPlainText[j]);
        }
        for (var j = 32; j < 64; j++) {
          r0.add(resultPermutationPlainText[j]);
        }

        // Interaksi ke-1
        var er0 = _allFunction.expansiWithTableExpansi(r0);
        var a1 = _allFunction.xOr(er0, ki[15].k);
        var b1 = _allFunction.subtitusiSBox(a1);
        var pb1 = _allFunction.permutationWithTablePBox(b1);
        var r1 = _allFunction.xOr(pb1, l0);
        var er1 = _allFunction.expansiWithTableExpansi(r1);

        // Interaksi ke-2
        var a2 = _allFunction.xOr(er1, ki[14].k);
        var b2 = _allFunction.subtitusiSBox(a2);
        var pb2 = _allFunction.permutationWithTablePBox(b2);
        var r2 = _allFunction.xOr(pb2, r0);
        var er2 = _allFunction.expansiWithTableExpansi(r2);

        // Interaksi ke-3
        var a3 = _allFunction.xOr(er2, ki[13].k);
        var b3 = _allFunction.subtitusiSBox(a3);
        var pb3 = _allFunction.permutationWithTablePBox(b3);
        var r3 = _allFunction.xOr(pb3, r1);
        var er3 = _allFunction.expansiWithTableExpansi(r3);

        // Interaksi ke-4
        var a4 = _allFunction.xOr(er3, ki[12].k);
        var b4 = _allFunction.subtitusiSBox(a4);
        var pb4 = _allFunction.permutationWithTablePBox(b4);
        var r4 = _allFunction.xOr(pb4, r2);
```

```
var er4 = _allFunction.expansiWithTableExpansi(r4);

// Interaksi ke-5
var a5 = _allFunction.xOr(er4, ki[11].k);
var b5 = _allFunction.subtitusiSBox(a5);
var pb5 = _allFunction.permutationWithTablePBox(b5);
var r5 = _allFunction.xOr(pb5, r3);
var er5 = _allFunction.expansiWithTableExpansi(r5);

// Interaksi ke-6
var a6 = _allFunction.xOr(er5, ki[10].k);
var b6 = _allFunction.subtitusiSBox(a6);
var pb6 = _allFunction.permutationWithTablePBox(b6);
var r6 = _allFunction.xOr(pb6, r4);
var er6 = _allFunction.expansiWithTableExpansi(r6);

// Interaksi ke-7
var a7 = _allFunction.xOr(er6, ki[9].k);
var b7 = _allFunction.subtitusiSBox(a7);
var pb7 = _allFunction.permutationWithTablePBox(b7);
var r7 = _allFunction.xOr(pb7, r5);
var er7 = _allFunction.expansiWithTableExpansi(r7);

// Interaksi ke-8
var a8 = _allFunction.xOr(er7, ki[8].k);
var b8 = _allFunction.subtitusiSBox(a8);
var pb8 = _allFunction.permutationWithTablePBox(b8);
var r8 = _allFunction.xOr(pb8, r6);
var er8 = _allFunction.expansiWithTableExpansi(r8);

// Interaksi ke-9
var a9 = _allFunction.xOr(er8, ki[7].k);
var b9 = _allFunction.subtitusiSBox(a9);
var pb9 = _allFunction.permutationWithTablePBox(b9);
var r9 = _allFunction.xOr(pb9, r7);
var er9 = _allFunction.expansiWithTableExpansi(r9);

// Interaksi ke-10
var a10 = _allFunction.xOr(er9, ki[6].k);
var b10 = _allFunction.subtitusiSBox(a10);
var pb10 = _allFunction.permutationWithTablePBox(b10);
var r10 = _allFunction.xOr(pb10, r8);
var er10 = _allFunction.expansiWithTableExpansi(r10);

// Interaksi ke-11
var a11 = _allFunction.xOr(er10, ki[5].k);
var b11 = _allFunction.subtitusiSBox(a11);
var pb11 = _allFunction.permutationWithTablePBox(b11);
var r11 = _allFunction.xOr(pb11, r9);
var er11 = _allFunction.expansiWithTableExpansi(r11);

// Interaksi ke-12
var a12 = _allFunction.xOr(er11, ki[4].k);
var b12 = _allFunction.subtitusiSBox(a12);
```

```
        var pb12 = _allFunction.permutationWithTablePBox(b12);
        var r12 = _allFunction.xOr(pb12, r10);
        var er12 = _allFunction.expansiWithTableExpansi(r12);

        // Interaksi ke-13
        var a13 = _allFunction.xOr(er12, ki[3].k);
        var b13 = _allFunction.subtitusiSBox(a13);
        var pb13 = _allFunction.permutationWithTablePBox(b13);
        var r13 = _allFunction.xOr(pb13, r11);
        var er13 = _allFunction.expansiWithTableExpansi(r13);

        // Interaksi ke-14
        var a14 = _allFunction.xOr(er13, ki[2].k);
        var b14 = _allFunction.subtitusiSBox(a14);
        var pb14 = _allFunction.permutationWithTablePBox(b14);
        var r14 = _allFunction.xOr(pb14, r12);
        var er14 = _allFunction.expansiWithTableExpansi(r14);

        // Interaksi ke-15
        var a15 = _allFunction.xOr(er14, ki[1].k);
        var b15 = _allFunction.subtitusiSBox(a15);
        var pb15 = _allFunction.permutationWithTablePBox(b15);
        var r15 = _allFunction.xOr(pb15, r13);
        var er15 = _allFunction.expansiWithTableExpansi(r15);

        // Interaksi ke-16
        var a16 = _allFunction.xOr(er15, ki[0].k);
        var b16 = _allFunction.subtitusiSBox(a16);
        var pb16 = _allFunction.permutationWithTablePBox(b16);
        var r16 = _allFunction.xOr(pb16, r14);
        var er16 = _allFunction.expansiWithTableExpansi(r16);

        var joinR16L16 = List<String>();
        for (var j = 0; j < r16.length; j++) {
          joinR16L16.add(r16[j]);
        }
        for (var j = 0; j < r15.length; j++) {
          joinR16L16.add(r15[j]);
        }
        var resultAll =
_allFunction.permutationWithTableIP1(joinR16L16);
        temp = _allFunction.binaryToString(resultAll);
      }
      _hasilDecrypt = _hasilDecrypt + temp;
    }
    return _hasilDecrypt;
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: SingleChildScrollView(
        child: Container(
          width: MediaQuery.of(context).size.width,
```

```
          padding: EdgeInsets.only(top: 10.0, left: 30.0, right:
30.0),
          child: Column(
            mainAxisAlignment: MainAxisAlignment.start,
            crossAxisAlignment: CrossAxisAlignment.start,
            children: <Widget>[
              Container(
                child: Column(
                  mainAxisAlignment: MainAxisAlignment.start,
                  crossAxisAlignment: CrossAxisAlignment.start,
                  children: <Widget>[
                    SizedBox(height: 18.0,),
                    Text(
                      "Encrypt",
                      style: TextStyle(
                        fontSize: 18.0,
                        fontWeight: FontWeight.w500
                      ),
                    ),
                    SizedBox(height: 18.0,),
                    Container(
                      width: MediaQuery.of(context).size.width,
                      padding: EdgeInsets.all(8.0),
                      decoration: BoxDecoration(
                        borderRadius: BorderRadius.circular(10.0),
                        color: Colors.grey[200],
                        boxShadow: [
                          BoxShadow(
                            color: Colors.black12,
                            offset: Offset(5.0, 3.0),
                            blurRadius: 20.0,
                          ),
                        ],
                        border: Border.all(
                          color: Colors.grey,
                          width: 0.8
                        )
                      ),
                      child: Column(
                        mainAxisAlignment:
MainAxisAlignment.start,
                        crossAxisAlignment:
CrossAxisAlignment.start,
                        children: <Widget>[
                          Text("Gambar"),
                          (_imageLampiranWA == null) ?
                          InkWell(
                            child: IgnorePointer(
                              child: TextFormField(
                                readOnly: true,
                                decoration: InputDecoration(
                                  prefixIcon: Icon(Icons.camera),
                                  hintText: "Pilih gambar"
                                ),
```

```
                    ),
                  ),
                  onTap: () {
                    _openImages();
                  },
                ) :
                Container(
                  child: Column(
                    mainAxisAlignment:
MainAxisAlignment.start,
                    crossAxisAlignment:
CrossAxisAlignment.end,
                    children: <Widget>[
                      Image.memory(_fotoLampiranWA),
                      SizedBox(height: 8.0,),
                      Container(
                        child: GestureDetector(
                          child: Text(
                            "Ubah Foto Bukti Visit",
                            style: TextStyle(color:
Colors.grey),
                          ),
                          onTap: () {
                            _openImages();
                          },
                        ),
                      )
                    ],
                  ),
                ),
                SizedBox(height: 14.0,),
                Text("Key"),
                TextFormField(
                  controller: _keyController,
                  maxLength: 8,
                  decoration: InputDecoration(
                    hintText: "Input kunci",
                  ),
                ),
                SizedBox(height: 14.0,),
                Padding(
                  padding: const EdgeInsets.all(8.0),
                  child: Container(
                    margin: EdgeInsets.only(bottom:
30.0),
                    width:
MediaQuery.of(context).size.width,
                    child: RaisedButton(
                      elevation: 8.0,
                      padding: const
EdgeInsets.all(15.0),
                      color: Colors.blue[200],
                      child: Text(
                        "ENCRYPT",
```

```
                                    style: TextStyle(
                                      color: Colors.white,
                                      fontSize: 18.0,
                                      fontWeight: FontWeight.bold
                                    ),
                                  ),
                                  shape: RoundedRectangleBorder(
                                    borderRadius:
BorderRadius.circular(8.0)
                                  ),
                                  onPressed: () async {
                                    print(DateTime.now());
                                    _resultEncryptHex =
_encrypt(_imageLampiranWA, _keyController.text);
                                    print(DateTime.now());
                                    print("PANJANG = $
{_resultEncryptHex.length}");
                                    setState(() {
                                      _cipherTextController.text =
_resultEncryptHex;
                                    });
                                  },
                                ),
                              ),
                            ),
                          ),
                        ],
                      ),
                    )
                  ],
                ),
              ),
              Container(
                child: Column(
                  mainAxisAlignment: MainAxisAlignment.start,
                  crossAxisAlignment: CrossAxisAlignment.start,
                  children: <Widget>[
                    SizedBox(height: 18.0,),
                    Text(
                      "Decrypt",
                      style: TextStyle(
                        fontSize: 18.0,
                        fontWeight: FontWeight.w500
                      ),
                    ),
                    SizedBox(height: 18.0,),
                    Container(
                      width: MediaQuery.of(context).size.width,
                      padding: EdgeInsets.all(8.0),
                      decoration: BoxDecoration(
                        borderRadius: BorderRadius.circular(10.0),
                        color: Colors.grey[200],
                        boxShadow: [
                          BoxShadow(
                            color: Colors.black12,
```

```
                        offset: Offset(5.0, 3.0),
                        blurRadius: 20.0,
                      ),
                    ],
                    border: Border.all(
                      color: Colors.grey,
                      width: 0.8
                    )
                  ),
                  child: Column(
                    mainAxisAlignment:
MainAxisAlignment.start,
                    crossAxisAlignment:
CrossAxisAlignment.start,
                    children: <Widget>[
                      Text("Cipher Text"),
                      TextFormField(
                        controller: _cipherTextController,
                        decoration: InputDecoration(
                          hintText: "Input cipher text"
                        ),
                      ),
                      SizedBox(height: 14.0,),
                      Text("Key"),
                      TextFormField(
                        controller: _keyController,
                        maxLength: 8,
                        decoration: InputDecoration(
                          hintText: "Input kunci",
                        ),
                      ),
                      SizedBox(height: 14.0,),
                      Padding(
                        padding: const EdgeInsets.all(8.0),
                        child: Container(
                          margin: EdgeInsets.only(bottom:
30.0),

                          width:
MediaQuery.of(context).size.width,
                          child: RaisedButton(
                            elevation: 8.0,
                            padding: const
EdgeInsets.all(15.0),

                            color: Colors.blue[200],
                            child: Text(
                              "DECRYPT",
                              style: TextStyle(
                                color: Colors.white,
                                fontSize: 18.0,
                                fontWeight: FontWeight.bold
                              ),
                            ),
                            shape: RoundedRectangleBorder(
```

```
                                    borderRadius:
BorderRadius.circular(8.0)
                                ),
                                onPressed: () async {
                                  print(DateTime.now());
                                  _resultDecrypt =
_decrypt(_cipherTextController.text, _keyController.text);
                                  print(DateTime.now());
                                  print("PANJANG DECRYPT = $
{_resultDecrypt.length}");
                                  setState(() {
                                    _hasilJadi =
base64Decode(_resultDecrypt);
                                  });
                                },
                              ),
                            ),
                          ),
                          (_hasilJadi != null) ?
                          Container(
                            child: Column(
                              mainAxisAlignment:
MainAxisAlignment.start,
                              crossAxisAlignment:
CrossAxisAlignment.end,
                              children: <Widget>[
                                Image.memory(_hasilJadi),
                                SizedBox(height: 8.0,),
                              ],
                            ),
                          ) :
                          Container(),
                        ],
                      ),
                    ),
                    SizedBox(height: 20.0,),
                  ],
                ),
              )
            ],
          ),
        ),
      ),
    );
  }
}
```

TabBar2.dart

```
import 'dart:convert';
import 'dart:typed_data';
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';
import 'package:flutter_des/flutter_des.dart';
```

```
import 'package:multi_image_picker/multi_image_picker.dart';

class TabBar2 extends StatefulWidget {
  @override
  _TabBar2State createState() => _TabBar2State();
}

class _TabBar2State extends State<TabBar2> {

  var _cipherTextController = TextEditingController();
  var _keyController = TextEditingController();
  Uint8List _fotoLampiranWA;
  Uint8List _hasilJadi;
  var _imageLampiranWA;

  var _resultEncryptHex = "";
  var _resultDecrypt = "";

  @override
  void initState() {
    super.initState();
  }

  Future _openImages() async {
    String base64Image;
    try {
      List<Asset> resultList = await MultiImagePicker.pickImages(
        maxImages: 1,
      );

      // Tambahan
      ByteData byteData = await resultList[0].getByteData(quality:
75);
      final buffer = byteData.buffer;
      var list = buffer.asUint8List(byteData.offsetInBytes,
byteData.lengthInBytes);
      base64Image = base64Encode(list);
      setState(() {
        _imageLampiranWA = base64Image;
        _fotoLampiranWA = base64Decode(base64Image);
      });
      print("BASE 64 = ${_imageLampiranWA.length}");
    }
    catch (e) {
      print(e);
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: SingleChildScrollView(
        child: Container(
          width: MediaQuery.of(context).size.width,
```

```
          padding: EdgeInsets.only(top: 10.0, left: 30.0, right:
30.0),
          child: Column(
            mainAxisAlignment: MainAxisAlignment.start,
            crossAxisAlignment: CrossAxisAlignment.start,
            children: <Widget>[
              Container(
                child: Column(
                  mainAxisAlignment: MainAxisAlignment.start,
                  crossAxisAlignment: CrossAxisAlignment.start,
                  children: <Widget>[
                    SizedBox(height: 18.0,),
                    Text(
                      "Encrypt",
                      style: TextStyle(
                        fontSize: 18.0,
                        fontWeight: FontWeight.w500
                      ),
                    ),
                    SizedBox(height: 18.0,),
                    Container(
                      width: MediaQuery.of(context).size.width,
                      padding: EdgeInsets.all(8.0),
                      decoration: BoxDecoration(
                        borderRadius: BorderRadius.circular(10.0),
                        color: Colors.grey[200],
                        boxShadow: [
                          BoxShadow(
                            color: Colors.black12,
                            offset: Offset(5.0, 3.0),
                            blurRadius: 20.0,
                          ),
                        ],
                        border: Border.all(
                          color: Colors.grey,
                          width: 0.8
                        )
                      ),
                      child: Column(
                        mainAxisAlignment:
MainAxisAlignment.start,
                        crossAxisAlignment:
CrossAxisAlignment.start,
                        children: <Widget>[
                          Text("Gambar"),
                          (_imageLampiranWA == null) ?
                          InkWell(
                            child: IgnorePointer(
                              child: TextFormField(
                                readOnly: true,
                                decoration: InputDecoration(
                                  prefixIcon: Icon(Icons.camera),
                                  hintText: "Pilih gambar"
                                ),
```

```
                                ),
                              ),
                              onTap: () {
                                _openImages();
                                //
_openImages("lampiranPerjalanan");
                              },
                            ) :
                            Container(
                              child: Column(
                                mainAxisAlignment:
MainAxisAlignment.start,
                                crossAxisAlignment:
CrossAxisAlignment.end,
                                children: <Widget>[
                                  Image.memory(_fotoLampiranWA),
                                  SizedBox(height: 8.0,),
                                  Container(
                                    child: GestureDetector(
                                      child: Text(
                                        "Ubah Foto Bukti Visit",
                                        style: TextStyle(color:
Colors.grey),
                                      ),
                                      onTap: () {
                                        _openImages();
                                      },
                                    ),
                                  )
                                ],
                              ),
                            ),
                            SizedBox(height: 14.0,),
                            Text("Key"),
                            TextFormField(
                              controller: _keyController,
                              maxLength: 8,
                              decoration: InputDecoration(
                                hintText: "Input kunci",
                              ),
                            ),
                            SizedBox(height: 14.0,),
                            Padding(
                              padding: const EdgeInsets.all(8.0),
                              child: Container(
                                margin: EdgeInsets.only(bottom:
30.0),
                                width:
MediaQuery.of(context).size.width,
                                child: RaisedButton(
                                  elevation: 8.0,
                                  padding: const
EdgeInsets.all(15.0),
                                  color: Colors.blue[200],
```

```
                                child: Text(
                                  "ENCRYPT",
                                  style: TextStyle(
                                    color: Colors.white,
                                    fontSize: 18.0,
                                    fontWeight: FontWeight.bold
                                  ),
                                ),
                                shape: RoundedRectangleBorder(
                                  borderRadius:
BorderRadius.circular(8.0)
                                ),
                                onPressed: () async {
                                  print(DateTime.now());
                                  var enkrip = await
FlutterDes.encryptToHex(_imageLampiranWA, _keyController.text, iv:
"12345678");

                                  print(DateTime.now());
                                  print("PANJANG = $
{enkrip.length}");

                                  setState(() {
                                    _cipherTextController.text =
enkrip;

                                  });
                                },
                              ),
                            ),
                          ),
                          (_resultEncryptHex != "") ?
                          Container(
                            child: Column(
                              mainAxisAlignment:
MainAxisAlignment.start,
                              crossAxisAlignment:
CrossAxisAlignment.start,
                              children: <Widget>[
                                Text(
                                  "Result Encrypt",
                                  style: TextStyle(
                                    fontWeight: FontWeight.w700,
                                  ),
                                ),
                                SelectableText(
                                  _resultEncryptHex,
                                  style: TextStyle(
                                    fontWeight: FontWeight.w700,
                                    fontSize: 20.0
                                  ),
                                )
                              ],
                            ),
                          ) :
                          Container()
                        ],
```

```
                ),
              )
            ],
          ),
        ),
        Container(
          child: Column(
            mainAxisAlignment: MainAxisAlignment.start,
            crossAxisAlignment: CrossAxisAlignment.start,
            children: <Widget>[
              SizedBox(height: 18.0,),
              Text(
                "Decrypt",
                style: TextStyle(
                  fontSize: 18.0,
                  fontWeight: FontWeight.w500
                ),
              ),
              SizedBox(height: 18.0,),
              Container(
                width: MediaQuery.of(context).size.width,
                padding: EdgeInsets.all(8.0),
                decoration: BoxDecoration(
                  borderRadius: BorderRadius.circular(10.0),
                  color: Colors.grey[200],
                  boxShadow: [
                    BoxShadow(
                      color: Colors.black12,
                      offset: Offset(5.0, 3.0),
                      blurRadius: 20.0,
                    ),
                  ],
                  border: Border.all(
                    color: Colors.grey,
                    width: 0.8
                  )
                ),
                child: Column(
                  mainAxisAlignment:
MainAxisAlignment.start,
                  crossAxisAlignment:
CrossAxisAlignment.start,
                  children: <Widget>[
                    Text("Cipher Text"),
                    TextFormField(
                      controller: _cipherTextController,
                      decoration: InputDecoration(
                        hintText: "Input cipher text"
                      ),
                    ),
                    SizedBox(height: 14.0,),
                    Text("Key"),
                    TextFormField(
                      controller: _keyController,
```

```
                maxLength: 8,
                decoration: InputDecoration(
                  hintText: "Input kunci",
                ),
              ),
              SizedBox(height: 14.0,),
              Padding(
                padding: const EdgeInsets.all(8.0),
                child: Container(
                  margin: EdgeInsets.only(bottom:
30.0),
                  width:
MediaQuery.of(context).size.width,
                  child: RaisedButton(
                    elevation: 8.0,
                    padding: const
EdgeInsets.all(15.0),
                    color: Colors.blue[200],
                    child: Text(
                      "DECRYPT",
                      style: TextStyle(
                        color: Colors.white,
                        fontSize: 18.0,
                        fontWeight: FontWeight.bold
                      ),
                    ),
                    shape: RoundedRectangleBorder(
                      borderRadius:
BorderRadius.circular(8.0)
                    ),
                    onPressed: () async {
                      print(DateTime.now());
                      var dekrip = await
FlutterDes.decryptFromHex(_cipherTextController.text,
_keyController.text, iv: "12345678");
                      print(DateTime.now());
                      print("PANJANG = $
{dekrip.length}");
                      setState(() {
                        _hasilJadi =
base64Decode(dekrip);
                      });
                    },
                  ),
                ),
              ),
              (_hasilJadi != null) ?
              Container(
                child: Column(
                  mainAxisAlignment:
MainAxisAlignment.start,
                  crossAxisAlignment:
CrossAxisAlignment.end,
                  children: <Widget>[
```

```
                                Image.memory(_hasilJadi),
                                SizedBox(height: 8.0,),
                              ],
                            ),
                          ) :
                          Container(),
                        ],
                      ),
                    ),
                    SizedBox(height: 20.0,),
                  ],
                ),
              )
            ],
          ),
        ),
      ),
    );
  }
}
```

AllFunction.dart

```
// ==========================================================
// ||======================================================||
// ||                                                      ||
// ||                      ALL TABLE                       ||
// ||                                                      ||
// ||======================================================||
// ==========================================================

class AllTable {
  List<int> tableIP() {
    var value = List<int>();
    value = [
      58, 50, 42, 34, 26, 18, 10, 2,
      60, 52, 44, 36, 28, 20, 12, 4,
      62, 54, 46, 38, 30, 22, 14, 6,
      64, 56, 48, 40, 32, 24, 16, 8,
      57, 49, 41, 33, 25, 17, 9, 1,
      59, 51, 43, 35, 27, 19, 11, 3,
      61, 53, 45, 37, 29, 21, 13, 5,
      63, 55, 47, 39, 31, 23, 15, 7,
    ];
    return value;
  }

  List<int> tablePC1() {
    var value = List<int>();
    value = [
      57, 49, 41, 33, 25, 17, 9,
      1, 58, 50, 42, 34, 26, 18,
      10, 2, 59, 51, 43, 35, 27,
      19, 11, 3, 60, 52, 44, 36,
```

```
      63, 55, 47, 39, 31, 23, 15,
      7, 62, 54, 46, 38, 30, 22,
      14, 6, 61, 53, 45, 37, 29,
      21, 13, 5, 28, 20, 12, 4,
    ];
    return value;
  }

  List<int> tablePC2() {
    var value = List<int>();
    value = [
      14, 17, 11, 24, 1, 5,
      3, 28, 15, 6, 21, 10,
      23, 19, 12, 4, 26, 8,
      16, 7, 27, 20, 13, 2,
      41, 52, 31, 37, 47, 55,
      30, 40, 51, 45, 33, 48,
      44, 49, 39, 56, 34, 53,
      46, 42, 50, 36, 29, 32,
    ];
    return value;
  }

  List<int> tableExpansi() {
    var value = List<int>();
    value = [
      32, 1, 2, 3, 4, 5,
      4, 5, 6, 7, 8, 9,
      8, 9, 10, 11, 12, 13,
      12, 13, 14, 15, 16, 17,
      16, 17, 18, 19, 20, 21,
      20, 21, 22, 23, 24, 25,
      24, 25, 26, 27, 28, 29,
      28, 29, 30, 31, 32, 1,
    ];
    return value;
  }

  List<String> row() {
    var value = List<String>();
    value = [
      "00", "01", "10", "11",
    ];
    return value;
  }

  List<String> column() {
    var value = List<String>();
    value = [
      "0000", "0001", "0010", "0011", "0100", "0101", "0110",
"0111",
      "1000", "1001", "1010", "1011", "1100", "1101", "1110",
"1111",
    ];
```

```
    return value;
  }

  List<List<List<String>>> tableSBox() {
    var value = List<List<List<String>>>();
    value = [
      // S-Box 1
      [
        ["1110", "0100", "1101", "0001", "0010", "1111", "1011",
"1000", "0011", "1010", "0110", "1100", "0101", "1001", "0000",
"0111"],
        ["0000", "1111", "0111", "0100", "1110", "0010", "1101",
"0001", "1010", "0110", "1100", "1011", "1001", "0101", "0011",
"1000"],
        ["0100", "0001", "1110", "1000", "1101", "0110", "0010",
"1011", "1111", "1100", "1001", "0111", "0011", "1010", "0101",
"0000"],
        ["1111", "1100", "1000", "0010", "0100", "1001", "0001",
"0111", "0101", "1011", "0011", "1110", "1010", "0000", "0110",
"1101"],
      ],
      // S-Box 2
      [
        ["1111", "0001", "1000", "1110", "0110", "1011", "0011",
"0100", "1001", "0111", "0010", "1101", "1100", "0000", "0101",
"1010"],
        ["0011", "1101", "0100", "0111", "1111", "0010", "1000",
"1110", "1100", "0000", "0001", "1010", "0110", "1001", "1011",
"0101"],
        ["0000", "1110", "0111", "1011", "1010", "0100", "1101",
"0001", "0101", "1000", "1100", "0110", "1001", "0011", "0010",
"1111"],
        ["1101", "1000", "1010", "0001", "0011", "1111", "0100",
"0010", "1011", "0110", "0111", "1100", "0000", "0101", "1110",
"1001"],
      ],
      // S-Box 3
      [
        ["1010", "0000", "1001", "1110", "0110", "0011", "1111",
"0101", "0001", "1101", "1100", "0111", "1011", "0100", "0010",
"1000"],
        ["1101", "0111", "0000", "1001", "0011", "0100", "0110",
"1010", "0010", "1000", "0101", "1110", "1100", "1011", "1111",
"0001"],
        ["1101", "0110", "0100", "1001", "1000", "1111", "0011",
"0000", "1011", "0001", "0010", "1100", "0101", "1010", "1110",
"0111"],
        ["0001", "1010", "1101", "0000", "0110", "1001", "1000",
"0111", "0100", "1111", "1110", "0011", "1011", "0101", "0010",
"1100"],
      ],
      // S-Box 4
      [
```

```
        ["0111", "1101", "1110", "0011", "0000", "0110", "1001",
"1010", "0001", "0010", "1000", "0101", "1011", "1100", "0100",
"1111"],
        ["1101", "1000", "1011", "0101", "0110", "1111", "0000",
"0011", "0100", "0111", "0010", "1100", "0001", "1010", "1110",
"1001"],
        ["1010", "0110", "1001", "0000", "1100", "1011", "0111",
"1101", "1111", "0001", "0011", "1110", "0101", "0010", "1000",
"0100"],
        ["0011", "1111", "0000", "0110", "1010", "0001", "1101",
"1000", "1001", "0100", "0101", "1011", "1100", "0111", "0010",
"1110"],
      ],
      // S-Box 5
      [
        ["0010", "1100", "0100", "0001", "0111", "1010", "1011",
"0110", "1000", "0101", "0011", "1111", "1101", "0000", "1110",
"1001"],
        ["1110", "1011", "0010", "1100", "0100", "0111", "1101",
"0001", "0101", "0000", "1111", "1010", "0011", "1001", "1000",
"0110"],
        ["0100", "0010", "0001", "1011", "1010", "1101", "0111",
"1000", "1111", "1001", "1100", "0101", "0110", "0011", "0000",
"1110"],
        ["1011", "1000", "1100", "0111", "0001", "1110", "0010",
"1101", "0110", "1111", "0000", "1001", "1010", "0100", "0101",
"0011"],
      ],
      // S-Box 6
      [
        ["1100", "0001", "1010", "1111", "1001", "0010", "0110",
"1000", "0000", "1101", "0011", "0100", "1110", "0111", "0101",
"1011"],
        ["1010", "1111", "0100", "0010", "0111", "1100", "1001",
"0101", "0110", "0001", "1101", "1110", "0000", "1011", "0011",
"1000"],
        ["1001", "1110", "1111", "0101", "0010", "1000", "1100",
"0011", "0111", "0000", "0100", "1010", "0001", "1101", "1011",
"0110"],
        ["0100", "0011", "0010", "1100", "1001", "0101", "1111",
"1010", "1011", "1110", "0001", "0111", "0110", "0000", "1000",
"1101"],
      ],
      // S-Box 7
      [
        ["0100", "1011", "0010", "1110", "1111", "0000", "1000",
"1101", "0011", "1100", "1001", "0111", "0101", "1010", "0110",
"0001"],
        ["1101", "0000", "1011", "0111", "0100", "1001", "0001",
"1010", "1110", "0011", "0101", "1100", "0010", "1111", "1000",
"0110"],
        ["0001", "0100", "1011", "1101", "1100", "0011", "0111",
"1110", "1010", "1111", "0110", "1000", "0000", "0101", "1001",
"0010"],
```

```
        ["0110", "1011", "1101", "1000", "0001", "0100", "1010",
"0111", "1001", "0101", "0000", "1111", "1110", "0010", "0011",
"1100"],
      ],
      // S-Box 8
      [
        ["1101", "0010", "1000", "0100", "0110", "1111", "1011",
"0001", "1010", "1001", "0011", "1110", "0101", "0000", "1100",
"0111"],
        ["0001", "1111", "1101", "1000", "1010", "0011", "0111",
"0100", "1100", "0101", "0110", "1011", "0000", "1110", "1001",
"0010"],
        ["0111", "1011", "0100", "0001", "1001", "1100", "1110",
"0010", "0000", "0110", "1010", "1101", "1111", "0011", "0101",
"1000"],
        ["0010", "0001", "1110", "0111", "0100", "1010", "1000",
"1101", "1111", "1100", "1001", "0000", "0011", "0101", "0110",
"1011"],
      ]
    ];
    return value;
  }

  List<int> tablePBox() {
    var value = List<int>();
    value = [
      16, 7, 20, 21, 29, 12, 28, 17,
      1, 15, 23, 26, 5, 18, 31, 10,
      2, 8, 24, 14, 32, 27, 3, 9,
      19, 13, 30, 6, 22, 11, 4, 25,
    ];
    return value;
  }

  List<int> tableIP1() {
    var value = List<int>();
    value = [
      40, 8, 48, 16, 56, 24, 64, 32,
      39, 7, 47, 15, 55, 23, 63, 31,
      38, 6, 46, 14, 54, 22, 62, 30,
      37, 5, 45, 13, 53, 21, 61, 29,
      36, 4, 44, 12, 52, 20, 60, 28,
      35, 3, 43, 11, 51, 19, 59, 27,
      34, 2, 42, 10, 50, 18, 58, 26,
      33, 1, 41, 9, 49, 17, 57, 25,
    ];
    return value;
  }
}

// ==================================================
// ||================================================||
// ||                                                ||
// ||               ENCRYPT & DECRYPT                ||
```

```
// ||                                                  ||
// ||==================================================||
// ====================================================

class AllFunction {

  // Encrypt
  var _allTable = AllTable();

  var _char128 = List<String>();
  var _char256 = List<String>();
  var _plainText = List<String>();
  var _char100000 = List<String>();
  var _cipherText = List<String>();
  var _resultTo4Bit = List<String>();

  List<String> splitTo1024Char(String kata) {
    if (kata.length > 128) {
      var kataPertama = kata.substring(0, 128);
      _char128.add(kataPertama);
      var kataSisa = kata.substring(128, kata.length);
      splitTo1024Char(kataSisa);
    }
    else {
      var kataSisa = kata.substring(0, kata.length);
      _char128.add(kataSisa);
    }
    return _char128;
  }

  List<String> splitTo256Char(String kata) {
    if (kata.length > 800) {
      var kataPertama = kata.substring(0, 800);
      _char256.add(kataPertama);
      var kataSisa = kata.substring(800, kata.length);
      splitTo256Char(kataSisa);
    }
    else {
      var kataSisa = kata.substring(0, kata.length);
      _char256.add(kataSisa);
    }
    return _char256;
  }

  // Mengubah string menjadi 64 bit
  List<String> splitTo64Bit(String kata) {
    if (kata.length > 8) {
      var kataPertama = kata.substring(0, 8);
      _plainText.add(kataPertama);
      var kataSisa = kata.substring(8, kata.length);
      splitTo64Bit(kataSisa);
    }
    else {
      var kataSisa = kata.substring(0, kata.length);
```

```
    _plainText.add(kataSisa);
  }
  return _plainText;
}

// Mengubah string ke binary
List<String> stringToBinary(String kata) {
  String binary;
  var data = List<String>();
  // var base64 = "";
  var decimal = kata.codeUnits;
  for (var i = 0; i < kata.length; i++) {
    binary = decimal[i].toRadixString(2);
    while (binary.length <8) {
      binary = "0" + binary;
    }
    for (var j = 0; j < binary.length; j++) {
      data.add(binary.substring(j, j+1));
    }
    // data.add(binary);
    // base64 = base64 + binary;
    binary = "";
  }
  if (kata.length != 8) {
    for (var i = kata.length % 8; i < 8; i++) {
      for (var j = 0; j < 8; j++) {
        data.add("0");
      }
    }
  }
  return data;
}

// Permutasi plaintext dengan tabel IP
List<String> permutationWithTableIP(List<String> value) {
  var resultPermutation = List<String>();
  var tableIP = _allTable.tableIP();

  for (var i = 0; i < tableIP.length; i++) {
    resultPermutation.add(value[tableIP[i]-1]);
  }

  return resultPermutation;
}

// Permutasi key dengan tabel PC1
List<String> permutationWithTablePC1(List<String> value) {
  var resultPermutation = List<String>();
  var tablePC1 = _allTable.tablePC1();

  for (var i = 0; i < tablePC1.length; i++) {
    resultPermutation.add(value[tablePC1[i]-1]);
  }
```

```
  return resultPermutation;
}

// Perputaran sebanyak 16x
List<String> leftShifting(List<String> value, int shifting) {
  var _resultLeftShifting = List<String>();
  // _resultLeftShifting = value;
  for (var i = 0; i < value.length; i++) {
    _resultLeftShifting.add(value[i]);
  }

  for (var i = 0; i < shifting; i++) {
    _resultLeftShifting.add(_resultLeftShifting[0]);
    _resultLeftShifting.removeAt(0);
  }

  return _resultLeftShifting;
}

// Permutasi hasil perputaran dengan tabel PC2
List<String> permutationWithTablePC2(List<String> value) {
  var resultPermutation = List<String>();
  var tablePC2 = _allTable.tablePC2();

  for (var i = 0; i < tablePC2.length; i++) {
    resultPermutation.add(value[tablePC2[i]-1]);
  }

  return resultPermutation;
}

List<String> expansiWithTableExpansi(List<String> value) {
  var resultExpansi = List<String>();
  var tableExpansi = _allTable.tableExpansi();

  for (var i = 0; i < tableExpansi.length; i++) {
    resultExpansi.add(value[tableExpansi[i]-1]);
  }

  return resultExpansi;
}

List<String> xOr(List<String> value1, List<String> value2) {
  var data1 = List<String>();
  var data2 = List<String>();
  var resultxOR = List<String>();

  for (var i = 0; i < value1.length; i++) {
    data1.add(value1[i]);
  }
  for (var i = 0; i < value2.length; i++) {
    data2.add(value2[i]);
  }
```

```
  for (var i = 0; i < value1.length; i++) {
    if (data1[i] == data2[i]) {
      resultxOR.add("0");
    }
    else {
      resultxOR.add("1");
    }
  }

  return resultxOR;
}

List<String> subtitusiSBox(List<String> value) {
  var resultSubsitusi = List<String>();
  var row = _allTable.row();
  var column = _allTable.column();
  var sBox = _allTable.tableSBox();
  var s = List<S>();
  var s1 = List<String>();
  var s2 = List<String>();
  var s3 = List<String>();
  var s4 = List<String>();
  var s5 = List<String>();
  var s6 = List<String>();
  var s7 = List<String>();
  var s8 = List<String>();
  for (var i = 0; i < value.length; i++) {
    if (i < 6) {
      s1.add(value[i]);
    }
    else if (i < 12) {
      s2.add(value[i]);
    }
    else if (i < 18) {
      s3.add(value[i]);
    }
    else if (i < 24) {
      s4.add(value[i]);
    }
    else if (i < 30) {
      s5.add(value[i]);
    }
    else if (i < 36) {
      s6.add(value[i]);
    }
    else if (i < 42) {
      s7.add(value[i]);
    }
    else if (i < 48) {
      s8.add(value[i]);
    }
  }

  s.add(S(s1));
```

```
    s.add(S(s2));
    s.add(S(s3));
    s.add(S(s4));
    s.add(S(s5));
    s.add(S(s6));
    s.add(S(s7));
    s.add(S(s8));

    for (var i = 0; i < s.length; i++) {
      var baris = "";
      var kolom = "";

      baris = s[i].s[0] + s[i].s[5];
      kolom = s[i].s[1] + s[i].s[2] + s[i].s[3] + s[i].s[4];

      for (var j = 0; j < row.length; j++) {
        if (baris == row[j]) {
          for (var k = 0; k < column.length; k++) {
            if (kolom == column[k]) {
              for (var l = 0; l < sBox[i][j][k].length; l++) {
                resultSubsitusi.add(sBox[i][j][k].substring(l,
l+1));
              }
            }
          }
        }
      }
    }

    return resultSubsitusi;
  }

  List<String> permutationWithTablePBox(List<String> value) {
    var resultPermutation = List<String>();
    var tablePBox = _allTable.tablePBox();

    for (var i = 0; i < tablePBox.length; i++) {
      resultPermutation.add(value[tablePBox[i]-1]);
    }

    return resultPermutation;
  }

  List<String> permutationWithTableIP1(List<String> value) {
    var resultPermutation = List<String>();
    var tableIP1 = _allTable.tableIP1();

    for (var i = 0; i < tableIP1.length; i++) {
      resultPermutation.add(value[tableIP1[i]-1]);
    }

    return resultPermutation;
  }
```

```
List<String> splitTo4Bit(List<String> value) {
  var _temp = List<String>();
  for (var i = 0; i < value.length; i++) {
    _temp.add(value[i]);
  }
  var _binary = "";
  if (_temp.length > 4) {
    for (var i = 0; i < 4; i++) {
      _binary = _binary + _temp[0];
      _temp.removeAt(0);
    }
    _resultTo4Bit.add(_binary);
    splitTo4Bit(_temp);
  }
  else {
    for (var i = 0; i < 4; i++) {
      _binary = _binary + _temp[i];
    }
    _resultTo4Bit.add(_binary);
  }

  return _resultTo4Bit;
}

String binaryToHex(List<String> value) {
  var _hex = "";
  for (var i = 0; i < value.length; i++) {
    switch (value[i]) {
      case "0000":
        _hex = _hex + "0";
        break;
      case "0001":
        _hex = _hex + "1";
        break;
      case "0010":
        _hex = _hex + "2";
        break;
      case "0011":
        _hex = _hex + "3";
        break;
      case "0100":
        _hex = _hex + "4";
        break;
      case "0101":
        _hex = _hex + "5";
        break;
      case "0110":
        _hex = _hex + "6";
        break;
      case "0111":
        _hex = _hex + "7";
        break;
      case "1000":
        _hex = _hex + "8";
```

```
        break;
      case "1001":
        _hex = _hex + "9";
        break;
      case "1010":
        _hex = _hex + "A";
        break;
      case "1011":
        _hex = _hex + "B";
        break;
      case "1100":
        _hex = _hex + "C";
        break;
      case "1101":
        _hex = _hex + "D";
        break;
      case "1110":
        _hex = _hex + "E";
        break;
      case "1111":
        _hex = _hex + "F";
        break;
      default:
    }
  }
  return _hex;
}

List<String> splitTo100000Char(String kata) {
  if (kata.length > 100000) {
    var kataPertama = kata.substring(0, 100000);
    _char100000.add(kataPertama);
    var kataSisa = kata.substring(100000, kata.length);
    splitTo100000Char(kataSisa);
  }
  else {
    var kataSisa = kata.substring(0, kata.length);
    _char100000.add(kataSisa);
  }
  return _char100000;
}

List<String> splitTo128Bit(String kata) {
  if (kata.length > 16) {
    var kataPertama = kata.substring(0, 16);
    _cipherText.add(kataPertama);
    var kataSisa = kata.substring(16, kata.length);
    splitTo128Bit(kataSisa);
  }
  else {
    var kataSisa = kata.substring(0, kata.length);
    _cipherText.add(kataSisa);
  }
  return _cipherText;
```

```
}

List<String> hexToBinary(String cipherText) {
  var binary = List<String>();
  for (var i = 0; i < cipherText.length; i++) {
    switch (cipherText.substring(i, i+1)) {
      case "0":
        binary.addAll(["0", "0", "0", "0"]);
        break;
      case "1":
        binary.addAll(["0", "0", "0", "1"]);
        break;
      case "2":
        binary.addAll(["0", "0", "1", "0"]);
        break;
      case "3":
        binary.addAll(["0", "0", "1", "1"]);
        break;
      case "4":
        binary.addAll(["0", "1", "0", "0"]);
        break;
      case "5":
        binary.addAll(["0", "1", "0", "1"]);
        break;
      case "6":
        binary.addAll(["0", "1", "1", "0"]);
        break;
      case "7":
        binary.addAll(["0", "1", "1", "1"]);
        break;
      case "8":
        binary.addAll(["1", "0", "0", "0"]);
        break;
      case "9":
        binary.addAll(["1", "0", "0", "1"]);
        break;
      case "A":
        binary.addAll(["1", "0", "1", "0"]);
        break;
      case "B":
        binary.addAll(["1", "0", "1", "1"]);
        break;
      case "C":
        binary.addAll(["1", "1", "0", "0"]);
        break;
      case "D":
        binary.addAll(["1", "1", "0", "1"]);
        break;
      case "E":
        binary.addAll(["1", "1", "1", "0"]);
        break;
      case "F":
        binary.addAll(["1", "1", "1", "1"]);
        break;
```

```
        default:
      }
    }
    return binary;
  }

  var _resultString = "";
  String binaryToString(List<String> binary) {
    var temp = List<String>();
    var biner = "";
    for (var i = 0; i < binary.length; i++) {
      temp.add(binary[i]);
    }
    if (temp.length > 8) {
      for (var i = 0; i < 8; i++) {
        biner = biner + temp[0];
        temp.removeAt(0);
      }
      if (biner != "00000000") {
        var decimal = int.parse(biner, radix:
2).toRadixString(10);
        var result = String.fromCharCode(int.parse(decimal));
        _resultString = _resultString + result;
      }
      else {
        _resultString = _resultString + "";
      }
      binaryToString(temp);
    }
    else {
      for (var i = 0; i < temp.length; i++) {
        biner = biner + temp[i];
      }
      if (biner != "00000000") {
        var decimal = int.parse(biner, radix:
2).toRadixString(10);
        var result = String.fromCharCode(int.parse(decimal));
        _resultString = _resultString + result;
      }
      else {
        _resultString = _resultString + "";
      }
    }
    // var decimal = int.parse(biner, radix: 2).toRadixString(10);
    // var hasil = String.fromCharCode(int.parse(decimal));
    // print(hasil);
    return _resultString;
  }
}

  // ======================================================
  // ||====================================================||
  // ||                                                    ||
  // ||                      CATETAN                       ||
```

```
  // ||                                                  ||
  // ||==================================================||
  // ====================================================

  // Binary 8 bit to decimal
  // var a = int.parse("01001011", radix: 2).toRadixString(10);

  // Decimal to string
  // String.fromCharCode(118);



class CiDi {
  final List<String> ci;
  final List<String> di;

  CiDi(this.ci, this.di);
}

class K {
  final List<String> k;

  K(this.k);
}

class S {
  final List<String> s;

  S(this.s);
}
```

UNICHECK

Submission author:
16k10016 JANG, MICHAEL RUDY KURNIAWAN

Check ID:
15892135

Check date:
14.01.2020 12:57:29 GMT+0

Check type:
Doc vs Internet + Library

Report date:
15.01.2020 03:08:55 GMT+0

User ID:
29143

File name: 16.K1.0016_Jang Michael RK.doc

File ID: 20189444 Page count: 25 Word count: 4565 Character count: 26469 File size: 95.50 KB

1.6% Matches

Highest match: 0.35% with source https://onlinelibrary.wiley.com/doi/full/10.1002/ijop.12351

1.42% Internet Matches	93	Page 27
0.18% Library matches	29	Page 27

1.29% Quotes

Quotes	2	Page 28

No references found

0% Exclusions

No exclusions found

Replacement

No replaced characters found