BAB V

PENUTUP

Hasil uji-coba terhadap aplikasi program ini setelah jadi, serta meneliti dari segi keamanan data yang telah mengalami proses enkripsi dan proses kompresi maka dapat disimpulkan beberapa hal serta beberapa saran penunjang yang dapat digunakan bila ada yang ingin melakukan pengembangan lebih lanjut terhadap aplikasi ini.

5.1 Kesimpulan

a. Proses pengacakan yang di dapat dari hasil *compress encrypt* dapat menambah nilai keamanan data, perbandingan terhadap data yang tidak mengalami pengacakan.

b. Proses ini dapat mengurangi kegiatan *overbooking* yang dilakukan oleh oknum agen.

5.2 Saran

a. Dilakukan upaya kreatif untuk melakukan pengembangan dengan menyesuaikannya pada perkembangan algoritma-algoritma kriptografi terbaru.

b. Kedepannya agar keilmuan kriptografi dapat dipelajari dan dikembangkan sebagai bagian dari kurikulum perkuliahan, agar nantinya mahasiswa dapat berkreasi untuk menciptakan suatu algoritma kriptografi yang baru.

DAFTAR PUSTAKA

Fowler, M. (2004). UML Distiled. Edisi 3, Penerbit ANDI, Yogyakarta.

Kadir, Abdul. (2004). Dasar Pemograman Web Dinamis dengan JSP. Penerbit ANDI, Yogyakarta.

Kurniawan, Yusuf. Kriptografi, Keamanan Internet dan Jaringan Komunikasi. Informatika, Bandung.

Munawar. (2005). Pemodelan Visual dengan UML. Graha Ilmu, Yogyakarta.

Munir, Rinaldi. (2005). Algoritma Stream Chiper dengan Chaos, Institut Teknologi Bandung, Bandung.

Munir, Rinaldi. (2006). Kriptografi.cetakan pertama, Informatika, Bandung.

Penerbit Andi. (2003). Memahami Model Enkripsi dan Security Data. Penerbit Andi, Yogyakarta.

Sanjaya, Ridwan dan Purbo, Onno W. Membangun Web dengan JSP. PT. Elex Media Komputindo, Jakarta.

Wicaksono, Adi. (2002). Dasar-Dasar Pemograman Java 2. Penerbit PT. Elex Media Komputido, Jakarta

Widodo Budiharto. (2005). Panduan Lengkap Pemograman J2EE. Penerbit ANDI, Yogyakarta.

FileProcessor.java

```
01 package my.project;
02
03 import java.io.FileInputStream;
04 import java.io.FileOutputStream;
05 import java.io.IOException;
06 import java.io.InputStream;
07 import java.io.OutputStream;
08
09 import my.project.lzss.LZSSInputStream;
10 import my.project.lzss.LZSSOutputStream;
11 import my.project.scop.ScopInputStream;
12 import my.project.scop.ScopOutputStream;
13
14 public class FileProcessor {
15
16      private final static byte F_PACK_MAGIC[] = { 0x73, 0x6C, 0x68, 0x21 };
17      /* magic number for packed files */
18      private final static byte F_NOPACK_MAGIC[] = { 0x73, 0x6C, 0x68, 0x2E };
19
23      private FileProcessor(){
24      }
25
26      public static void compressEncrypt(String key, String fileInput, String    fileOutput) throws IOException{
27           InputStream in = new FileInputStream(fileInput);
28           OutputStream out= new FileOutputStream(fileOutput);
29           out.write(F_PACK_MAGIC);
30           out = new LZSSOutputStream(new ScopOutputStream(key.getBytes(), 0, key.length(), out));
31           byte b[] = new byte[512];
32           int i;
33           while ((i = in.read(b)) !=-1) {
34                out.write(b, 0, i);
35           }
36           out.close();
37           in.close();
38      }
39
40      public static void decryptDecompress(String key, String fileInput, String  fileOutput) throws IOException {
41           InputStream in = new FileInputStream(fileInput);
42           OutputStream out= new FileOutputStream(fileOutput);
43
44           byte magic[] = new byte[4];
45           in.read(magic);
46           int bad = 0;
47           for (int i= 2;i>=0; i--){
48                if (magic[i] != F_PACK_MAGIC[i])
49                     bad = 1;
50           }
51
52           if (bad == 0){
53                if (magic[3]  != F_PACK_MAGIC[3]){
54                     if (magic[3] == F_NOPACK_MAGIC[3]){
55                          bad = 2;
56                     } else {
57                          bad = 1;
58                     }
```

```
59                  }
60              }
61
62              if(bad== 1){
63                  throw new IOException("Error: " + fileInput + " - Not a packed file. (No magic)");
64              }
65              if (bad == 0){
66                  in = new LZSSInputStream(new ScopInputStream(key.getBytes(), 0, key.length(),  in));
67              }
68
69              byte b[]= new byte[512];
70              int i;
71              while ((i= in.read(b))!= -1) {
72                  out.write(b, 0, i);
73              }
74              out.close();
75              in.close();
76      }
77
78}
```

PassengerData.java

```
1 package my.project;
2
5 public class PassengerData {

        private String name;
        private String departure;
        private String destination;
        private String flightDate;
        private String ticketNumber;
        private String bookingNumber;
        private String fareBasis;

        public String getBookingNumber() {
            return bookingNumber;
        }
        public void setBookingNumber(String bookingNumber) {
            this.bookingNumber = bookingNumber;
        }
        public String getDeparture() {
            return departure;
        }
        public void setDeparture(String departure) {
            this.departure = departure;
        }
        public String getDestination() {
            return destination;
        }
        public void setDestination(String destination) {
            this.destination = destination;
        }
        public String getFareBasis() {
            return fareBasis;
```

```
    }
    public void setFareBasis(String fareBasis) {
        this.fareBasis = fareBasis;
    }
    public String getFlightDate() {
        return flightDate;
    }
    public void setFlightDate(String flightDate) {
        this.flightDate = flightDate;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getTicketNumber() {
        return ticketNumber;
    }
    public void setTicketNumber(String ticketNumber) {
        this.ticketNumber= ticketNumber;
    }


}
```

ProcessedFile.java

```
package my.project;

import java.io.Serializable;

public class ProcessedFile implements Serializable {

    private static final long serialVersionUID = 1L;

    private String fileNameAsal;

    private long fileSizeAsal;

    private String fileNameHasil;

    private long fileSizeHasil;

    private String processName;

    private boolean encrypt;

    public String getFileNameAsal() {
        return fileNameAsal;
    }

    public void setFileNameAsal(String fileNameAsal) {
        this.fileNameAsal = fileNameAsal;
    }
```

```
    public String getFileNameHasil() {
        return fileNameHasil;
    }

    public void setFileNameHasil(String fileNameHasil) {
        this.fileNameHasil = fileNameHasil;
    }

    public long getFileSizeAsal() {
        return fileSizeAsal;
    }

    public void setFileSizeAsal(long fileSizeAsal) {
        this.fileSizeAsal = fileSizeAsal;
    }

    public long getFileSizeHasil() {
        return fileSizeHasil;
    }

    public void setFileSizeHasil(long fileSizeHasil) {
        this.fileSizeHasil = fileSizeHasil;
    }

    public String getProcessName() {
        return processName;
    }

    public void setProcessName(String processName) {
        this.processName = processName;
    }

    public boolean isEncrypt() {
        return encrypt;
    }

    public void setEncrypt(boolean encrypt) {
        this.encrypt = encrypt;
    }

    public double getPersen() {
        return ((double)this.fileSizeHasil / (double)this.fileSizeAsal) * 100;
    }

}
```

FileProcessorActionBean.java

```
package my.project.action;

import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;

import javax.servlet.ServletOutputStream;
import javax.servlet.http.HttpServletResponse;
```

```
9
0 import my.project.FileProcessor;
1 import my.project.ProcessedFile;
2 import net.sourceforge.stripes.action.ActionBean;
3 import net.sourceforge.stripes.action.ActionBeanContext;
4 import net.sourceforge.stripes.action.DefaultHandler;
5 import net.sourceforge.stripes.action.FileBean;
6 import net.sourceforge.stripes.action.ForwardResolution;
7 import net.sourceforge.stripes.action.Resolution;
8 import net.sourceforge.stripes.action.StreamingResolution;
9 import net.sourceforge.stripes.validation.SimpleError;
0
1 public class FileProcessorActionBean implements ActionBean {
2
3       private ActionBeanContext ctx;
4
5       // file upload
6       private FileBean fileBean;
7
8       // password
9       private String password;
0
1       // processed file
2       private ProcessedFile processedFile;
3
4       // time processing
5       private double time;
6
7       // file to be downloaded
8       private String fileName;
9
0       public Resolution encrypt(){
1             String path      = ctx.getServletContext().getRealPath("/") + "upload/";
2             String fileInput  = fileBean.getFileName();
3             String fileOutput = fileInput.substring(0, fileInput.lastIndexOf('.'))+".rif";
4
5             processedFile  = new ProcessedFile();
6             processedFile.setFileNameAsal(fileBean.getFileName());
7             processedFile.setFileSizeAsal(fileBean.getSize());
8
9             double start= System.currentTimeMillis();
0             try {
1                   // save file
2                   fileBean.save(new File(path + fileInput));
3
4                   // compress & encrypt file
5                   FileProcessor.compressEncrypt(password, (path + fileInput), (path + fileOutput));
6
7                   File file = new File(path + fileOutput);
8                   processedFile.setFileNameHasil(fileOutput);
9                   processedFile.setFileSizeHasil(file.length());
0                   processedFile.setProcessName("Compress dan Encrypt");
1                   processedFile.setEncrypt(true);
2             } catch (IOException e) {
3                   ctx.getValidationErrors().addGlobalError(new SimpleError(e.getMessage()));
4                   return ctx.getSourcePageResolution();
5             }
```

```
        double end  = System.currentTimeMillis();
        time = (end - start)/1000;
        return new ForwardResolution("/result.jsp");
    }

    public Resolution decrypt(){
        String path     = ctx.getServletContext().getRealPath("/") + "upload/";
        String fileInput  = fileBean.getFileName();
        String fileOutput = fileInput.substring(0, fileInput.lastIndexOf('.'))+".txt";

        processedFile  = new ProcessedFile();
        processedFile.setFileNameAsal(fileBean.getFileName());
        processedFile.setFileSizeAsal(fileBean.getSize());

        double start = System.currentTimeMillis();
        try {
            // save file
            fileBean.save(new File(path + fileInput));

            // encrypt & decompress file
            FileProcessor.decryptDecompress(password, (path + fileInput), (path + fileOutput));

            File file = new File(path + fileOutput);
            processedFile.setFileNameHasil(fileOutput);
            processedFile.setFileSizeHasil(file.length());
            processedFile.setProcessName("Decrypt dan Decompress");
        } catch (IOException e) {
            ctx.getValidationErrors().addGlobalError(new SimpleError("please input correct password!"));
            return ctx.getSourcePageResolution();
        }

        double end  = System.currentTimeMillis();
        time = (end - start)/1000;
        return new ForwardResolution("/result.jsp");
    }

    @DefaultHandler
    public Resolution download(){
        final String path = ctx.getServletContext().getRealPath("/") + "upload/";
        final byte[] buf = new byte[1024];

        return new StreamingResolution("application/zip"){
            public void stream(HttpServletResponse response) throws IOException {
                File file = new File(path + fileName);
                response.setHeader("Content-Disposition", "attachment; filename=\\"+fileName+"\"");
                response.setHeader("Content-Length:", ""+file.length());
                ServletOutputStream out = response.getOutputStream();
                FileInputStream in = new FileInputStream(path + fileName);
                int numRead = 0;
                while ((numRead = in.read(buf)) >= 0) {
                    out.write(buf, 0, numRead);
                }
                in.close();
            }
        }.setFilename(fileName);
    }

    public double getTime() {
```

```
        return time;
    }

    public void setTime(double time) {
        this.time = time;
    }

    public FileBean getFileBean() {
        return fileBean;
    }

    public void setFileBean(FileBean fileBean) {
        this.fileBean = fileBean;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public ProcessedFile getProcessedFile() {
        return processedFile;
    }

    public void setProcessedFile(ProcessedFile processedFile) {
        this.processedFile = processedFile;
    }

    public String getFileName() {
        return fileName;
    }

    public void setFileName(String fileName) {
        this.fileName = fileName;
    }

    public ActionBeanContext getContext() {
        return ctx;
    }

    public void setContext(ActionBeanContext ctx) {
        this.ctx = ctx;
    }

}
```

PassengerActionBean.java

```
package my.project.action;

import java.io.DataOutputStream;
import java.io.File;
import java.io.FileInputStream;
```

```
import java.io.FileOutputStream;
import java.io.IOException;

import javax.servlet.ServletOutputStream;
import javax.servlet.http.HttpServletResponse;

import my.project.PassengerData;
import net.sourceforge.stripes.action.ActionBean;
import net.sourceforge.stripes.action.ActionBeanContext;
import net.sourceforge.stripes.action.Resolution;
import net.sourceforge.stripes.action.StreamingResolution;
import net.sourceforge.stripes.validation.SimpleError;

public class PassengerActionBean implements ActionBean {

    private ActionBeanContext ctx;

    private final static String LINE_FEED = System.getProperty("line.separator");

    // passenger data
    private PassengerData passenger;

    public Resolution proceed(){
        final byte[] buf = new byte[64];

        final String output   = buildOuput();
        final String path     = ctx.getServletContext().getRealPath("/") + "upload/";
        final String fileName = passenger.getTicketNumber()+".txt";
        final File file= new File(path + fileName);

        if(file.exists()){
            ctx.getValidationErrors().addGlobalError(new SimpleError("This ticket number already used."));
            return ctx.getSourcePageResolution();
        }

        try {
            FileOutputStream fos = new FileOutputStream(path + fileName);
            FileInputStream license = new FileInputStream(path + "LICENSE.txt");
            DataOutputStream dos= new DataOutputStream(fos);
            dos.writeBytes(output);
            while (true) {
                int bytesRead = license.read(buf);
                if(bytesRead ==-1)
                    break;
                dos.write(buf, 0, bytesRead);
            }

            dos.close();
            fos.close();
            license.close();
        } catch (IOException e) {
            ctx.getValidationErrors().addGlobalError(new SimpleError(e.getMessage()));
            return ctx.getSourcePageResolution();
        }

        return new StreamingResolution("plain/text"){
            public void stream(HttpServletResponse response) throws IOException {
                response.setHeader("Content-Disposition", "attachment; filename=\"+fileNam e+ "\");
```

```
                response.setHeader("Content-Length:", ""+file.length());
                ServletOutputStream out = response.getOutputStream();
                FileInputStream in = new FileInputStream(path + fileName);
                while (true) {
                        int bytesRead = in.read(buf);
                        if(bytesRead ==-1)
                                break;
                        out.write(buf, 0, bytesRead);
                }
                in.close();
            }
        }.setFilename(fileName);
    }

    private String buildOuput(){
        StringBuffer buf = new StringBuffer();
        buf.append("Data Personel Penumpang");
        buf.append(LINE_FEED);
        buf.append("Passenger Name\t\t:");
        buf.append(passenger.getName());
        buf.append(LINE_FEED);
        buf.append("Departure\t\t:");
        buf.append(passenger.getDeparture());
        buf.append(LINE_FEED);
        buf.append("Destination\t\t:");
        buf.append(passenger.getDestination());
        buf.append(LINE_FEED);
        buf.append("Date of Flight\t\t:");
        buf.append(passenger.getFlightDate());
        buf.append(LINE_FEED);
        buf.append("Ticket Number\t\t:");
        buf.append(passenger.getTicketNumber());
        buf.append(LINE_FEED);
        buf.append("Booking Number\t\t:");
        buf.append(passenger.getBookingNumber());
        buf.append(LINE_FEED);
        buf.append("Fare Basis\t\t:");
        buf.append(passenger.getFareBasis());
        buf.append(LINE_FEED);
        buf.append(LINE_FEED);
        return new String(buf);
    }

    public PassengerData getPassenger() {
        return passenger;
    }

    public void setPassenger(PassengerData passenger) {
        this.passenger = passenger;
    }

    public ActionBeanContext getContext() {
        return ctx;
    }

    public void setContext(ActionBeanContext ctx) {
        this.ctx = ctx;
    }
```

```
}
```

LZSSInputStream.java

```
package my.project.lzss;

/**************************************************
 ********** LZSS compression routines **********
 **************************************************

 This compression algorithm is based on the ideas of Lempel and Ziv,
 with the modifications suggested by Storer and Szymanski. The algorithm
 is based on the use of a ring buffer, which initially contains zeros.
 We read several characters from the file into the buffer, and then
 search the buffer for the longest string that matches the characters
 just read, and output the length and position of the match in the buffer.

 With a buffer size of 4096 bytes, the position can be encoded in 12
 bits  If we represent the match length in four bits, the <position,
 length> pair is two bytes long. If the longest match is no more than
 two characters, then we send just one character without encoding, and
 restart the process with the next letter. We must send one extra bit
 each time to tell the decoder whether we are sending a <position,
 length> pair or an unencoded character, and these flags are stored as
 an eight bit mask every eight items.

 This implementation uses binary trees to speed up the search for the
 longest match.

 Original code by Haruhiko Okumura, 4/6/1989.
 12-2-404 Green Heights, 580 Nagasawa, Yokosuka 239, Japan.

 Use, distribute, and modify this code freely.
 */

import java.io.FilterInputStream;
import java.io.IOException;
import java.io.InputStream;

public class LZSSInputStream extends FilterInputStream {

    final private static int N = 4096;

    final private static int F = 1 8 ; /*  */

    final private static int THRESHOLD = 2;

    /* for reading LZ files */
    int state; /* where have we got to? */

    int i, j, k, r, c;

    int flags;

    byte text_buf[];
```

```
public LZSSInputStream(InputStream i) {
        super(i);
        text_buf = new byte[N + F - 1];
}

/*
 * pack_read: Called by refill_buffer(). Unpacks from dat into buf, until
 * either EOF is reached or s bytes have been extracted. Returns the number
 * of bytes added to the buffer
 */

/* code stream */
/* bits flag , bit 1-quoted byte 0-packed string */
/* follows 8 codes, types determined by flagbits */
/* 1st code type is in the lowest flag bit */
/* packed string is encoded in 16 bits: */
/* 12 bits buffer position, 4 bits size */
/*
 * 1st byte: low part of pos. 2nd byte: high 4 bits = 4 high bits of pos.
 * low 4 bits = size - 2
 */

public final int read(byte buf[], int bufi, int s) throws IOException {
        int size = 0;
        if (state == 0) {
                r = N - F; /* fingbuffer pos? */
                flags = 0; /* zero flags */
        }
        while (s > 0) {
                if (state == 2) {
                        /* output encoded string from buffer*/
                        if (k <= j){
                                c = text_buf[(j + k) & (N - 1)];
                                text_buf[r++]= (byte) c;
                                buf[bufi++] = (byte) c;
                                r&= (N - 1);
                                k++;
                                --s;
                                size++;
                                continue;
                        }
                }
                /* test zda mame nacist dalsi flags byte */
                if (((flags >>>= 1) & 256) == 0) {
                        if ((c= in.read())==-1)
                                break;
                        flags = c | 0xFF00; /* uses higher byte to count eight */
                }
                if ((flags & 1)== 1){
                        /* quoted character */
                        if ((c = in.read())==-1)
                                break;
                        text_buf[r++] = (byte) c; /* vlozit do bufferu */
                        r&= N -1 ; /* korekce pozice v ring bufferu */
                        buf[bufi++]= (byte) c; /* vystup ven */
                        ++size;
                        --s;
```

```
                    state = 1;
                    continue;
                }
                /* encoded string */
                if ((j = in.read()) == -1)
                    break;
                if ((j = in.read()) ==-1)
                    break;
                i |= ((j & 0xF0) << 4); /* position */
                j = (j & 0x0F) + THRESHOLD; /* size */
                k = 0 ; /* output str. pos */
                state = 2;
            } /* while s>0 */

            if (size == 0)
                return -1;
            else
                return size;
        }

        public int read() throws IOException {
            byte b[] = new byte[1];
            int rc = read(b, 0, 1);
            if ( rc==-1 )
                return -1;
            byte z = b[0];
            return z & 0xFF;
        }

        public void close() throws IOException {
            text_buf = null;
            super.close();
            in = null;
        }

        /* non supported interface methods*/

        public boolean markSupported() {
            return false;
        }

        public long skip(long n) {
            return 0;
        }

        public int available() {
            return 0;
        }

        public void mark(int readlimit) {

        }

        public void reset() throws IOException {
            throw new IOException("Reset is not supported");
        }
    }
```

```
1 package my.project.lzss;
2
3 /**************************************************
4  ************ LZSS compression routines ************
5  **************************************************
6
7  This compression algorithm is based on the ideas of Lempel and Ziv,
8  with the modifications suggested by Storer and Szymanski. The algorithm
9  is based on the use of a ring buffer, which initially contains zeros.
0  We read several characters from the file into the buffer, and then
1  search the buffer for the longest string that matches the characters
2  just read, and output the length and position of the match in the buffer.
3
4  With a buffer size of 4096 bytes, the position can be encoded in 12
5  bits. If we represent the match length in four bits, the <position,
6  length> pair is two bytes long. If the longest match is no more than
7  two characters, then we send just one character without encoding, and
8  restart the process with the next letter. We must send one extra bit
9  each time to tell the decoder whether we are sending a <position,
0  length> pair or an unencoded character, and these flags are stored as
1  an eight bit mask every eight items.
2
3  This implementation uses binary trees to speed up the search for the
4  longest match.
5
6  Original code by Haruhiko Okumura, 4/6/1989.
7  12-2-404 Green Heights, 580 Nagasawa, Yokosuka 239, Japan.
8
9  Use, distribute, and modify this code freely.
0 */

import java.io.FilterOutputStream;
import java.io.IOException;
import java.io.OutputStream;

public class LZSSOutputStream extends FilterOutputStream {

    final private static int N = 4096;

    final private static int F = 18; /*  */

    final private static int THRESHOLD = 2;

    int state;

    int i, len, r, s;

    byte c;

    int last_match_length, code_buf_ptr;

    byte mask;

    byte code_buf[];
```

```
int match_position;

int match_length;

int lson[];

int rson[];

int dad[];

byte text_buf[];

public LZSSOutputStream(OutputStream ou) {
        super(ou);
        code_buf = new byte[17];
        lson = new int[N + 1];
        rson = new int[N + 257];
        dad = new int[N + 1];
        text_buf = new byte[N + F - 1];
        state = 0;
}

/*
 * pack_inittree: For i = 0 to N-1, rson[i] and lson[i] will be the right
 * and left children of node i. These nodes need not be initialized. Also,
 * dad[i] is the parent of node i. These are initialized to N, which stands
 * for 'not used.' For i = 0 to 255, rson[N+i+1] is the root of the tree for
 * strings that begin with character i. These are initialized to N. Note
 * there are 256 trees.
 */

private final void inittree() {
        int i;

        for (i = N + 1 ;i <= N + 256; i++)
                rson[i] = N;

        for (i = 0; i < N; i++)
                dad[i] = N;
}

/*
 * pack_insertnode: Inserts a string of length F, text_buf[r..r+F-1], into
 * one of the trees (text_buf[r]'th tree) and returns the longest-match
 * position and length via match_position and match_length. If match_length =
 * F, then removes the old node in favor of the new one, because the old one
 * will be deleted sooner. Noter plays double role, as tree node and
 * position in the buffer.
 */
private final void pack_insertnode(int r) {
        int i, p, cmp;
        // unsigned char *key;
        // unsigned char *text_buf = text_buf;

        cmp= 1;
        // key = &text_buf[r];
        p = N + 1 + (text_buf[r] & 0xFF);
        rson[r] = lson[r] = N;
```

```
        match_length = 0;

        for(;;) {

            if (cmp >= 0) {
                if (rson[p] != N)
                    p = rson[p];
                else {
                    rson[p] = r;
                    dad[r] = p;
                    return;
                }
            } else {
                if (lson[p] != N)
                    p = lson[p];
                else{
                    lson[p] = r;
                    dad[r] = p;
                    return;
                }
            }

            for ( i= 1; i < F ; i++)
                if ((cmp = (text_buf[r + i] & 0xff) - (text_buf[p + i]) & 0xFF) != 0)
                    break;

            if (i > match_length) {
                match_position = p;
                if ((match_length =i ) >= F)
                    break;
            }
        }

        dad[r] = dad[p];
        lson[r] = lson[p];
        rson[r] = rson[p];
        dad[lson[p]] = r;
        dad[rson[p]] = r;
        if (rson[dad[p]] == p)
            rson[dad[p]] = r;
        else
            lson[dad[p]] = r;
        dad[p] = N; /* remove p */
    }

    /*
     * pack_deletenode: Removes a node from a tree.
     */
    private final void pack_deletenode(int p) {
        int q;

        if (dad[p] == N)
            return; /* not in tree*/

        if (rson[p] == N)
            q = lson[p];
        else if (lson[p] == N)
            q = rson[p];
```

```
        else {
            q = lson[p];
            if (rson[q]!= N) {
                do {
                    q = rson[q];
                }while (rson[q]!= N);
                rson[dad[q]] = lson[q];
                dad[lson[q]] = dad[q];
                lson[q] = lson[p];
                dad[lson[p]]= q;
            }
            rson[q] = rson[p];
            dad[rson[p]]= q;
        }

        dad[q] = dad[p];
        if (rson[dad[p]] == p)
            rson[dad[p]] = q;
        else
            lson[dad[p]] = q;

        dad[p] = N;
    }

    /*
     * pack_write: Called by flush_buffer(). Packs size bytes from buf, using
     * the pack information contained in dat. Returns 0 on success.
     */
    private final void pack_write(int size, byte buf[], int bufi, boolean last)
            throws IOException {
        // System.out.println("Entering state="+state);
        boolean skipme = false;
        if (state == 0) {
            code_buf[0] = 0;
            /*
             * code_buf[1..16] saves eight units of code, and code_buf[0] works
             * as eight flags, "1" representing that the unit is an unencoded
             * letter (1 byte), "0" a position-and-length pair (2 bytes). Thus,
             * eight units require at most 16 bytes of code.
             */

            code_buf_ptr = mask = 1;

            s = 0;
            r = N - F;
            inittree();
            len = 0;
        } else if (state == 1)
            len++;

        if (state != 2) {
            while ((len < F) && (size > 0)) {
                // for (; (len < F) && (size > 0); len++) x
                text_buf[r + len] = buf[bufi++];
                if (--size == 0) {
                    if (!last) {
                        state = 1;
                        return;
```

```
                    }
                }
                pos1: len++;
            }

            if (len == 0)
                return;

            for (i = 1; i <= F; i++)
                pack_insertnode(r - i);
            /*
             * Insert the F strings, each of which begins with one or more
             * 'space' characters. Note the order in which these strings are
             * inserted. This way, degenerate trees will be less likely to
             * occur.
             */

            pack_insertnode(r);
            /*
             * Finally, insert the whole string just read. match_length and
             * match_position are set.
             */
        } /* state!=2 */
        else{
            skipme = true;
        }

        do {
            if (skipme == false) {
                if (match_length > len)
                    match_length = len; /* match_length may be long near the end */

                if (match_length <= THRESHOLD) {
                    match_length = 1; /* not long enough match: send one byte */
                    code_buf[0] |= mask; /* 'send one byte' flag */
                    code_buf[code_buf_ptr++] = text_buf[r] /* send uncoded */
                } else {
                    /*
                     * send position and length pair. Note match_length >
                     *THRESHOLD
                     */
                    code_buf[code_buf_ptr++] = (byte) match_position;
                    code_buf[code_buf_ptr++] = (byte) (((match_position >>> 4) & 0xF0) |
length - (THRESHOLD + 1)));
                }

                if ((mask <<= 1) == 0) { /* shift mask left one bit */
                    /* send at most 8 units of */
                    /* code together */
                    out.write(code_buf, 0, code_buf_ptr);
                    code_buf[0] = 0;
                    code_buf_ptr = mask = 1;
                }

                last_match_length = match_length;
                i = 0;
            } /* skipme */
```

```
        /* jak se dostat dovnitr? */
        /*
         * if(skipme==true && ((i < last_match_length) && (size > 0))==
         * false) { System.out.println("Can not get inside.... size="+size); //
         * skipme=false; }
         */
        for(;;) {
            // for (; (i < last_match_length) && (size > 0); i++)
            if (skipme == false) {
                if ((i >= last_match_length) || (size <= 0))
                    break;
                c = buf[bufi++];
                if (--size == 0) {
                    if (!last) {
                        state = 2;
                        return;
                    }
                }
            } else {
                skipme = false;/* System.out.println("Skip in") */
                ;
            }
            pos2: pack_deletenode(s); /* delete old strings and */
            text_buf[s] = c; /* read new bytes */
            if (s < F - 1)
                text_buf[s + N] = c;
            /*
             * if the position is near the end of buffer, extend the buffer
             * to make string comparison easier
             */
            s = (s + 1 ) & (N - 1);
            r = (r + 1 ) & (N - 1 ) ; /*
                                       * since this is a ring buffer,
                                       * increment the position modulo N
                                       */

            pack_insertnode(r); /*
                                 * register the string in text_buf[r..r+F-1]
                                 */
            i++;
        }

        while (i++ < last_match_length) { /* after the end of text, */
            pack_deletenode(s); /* no need to read, but */
            s = (s + 1 ) & (N - 1 );/* buffer may not be empty */
            r = (r + 1 ) & ( N - 1);
            if (--len != 0)
                pack_insertnode(r);
        }

    } while (len > 0); /* until length of string to be processed is zero*/

    if (code_buf_ptr > 1) { /* send remaining code */

        out.write(code_buf, 0, code_buf_ptr);
        return;
    }
    /* do not reset buffer */
```

```
        // state= 0;
    }

    /* interface methods */

    public void write(byte zz[], int ofs, int sz) throws IOException {
        pack_write(sz, zz, ofs, false);
    }

    public void flush() throws IOException {
        // System.out.println("Flush state="+state);
        pack_write(0, null, 0, true);
        super.flush();
    }

    public void close() throws IOException {

        if (code_buf == null)
            throw new IOException("LZSSOutputStream allready closed");
        try {
            flush();
            out.close();
        } catch (IOException err) {
            throw err;
        } finally {
            out= null;
            code_buf = null;
            lson = null;
            rson = null;
            dad = null;
            text_buf = null;
        }
    }

    protected final void finalize() throws Throwable {
        try {
            flush();
        } finally {
            super.finalize();
        }
    }

}
```

BinConverter.java

```
package my.project.scop;

public class BinConverter {
    public final static int byteArrayToInt(byte[] buf, int ofs) {
        return (buf[ofs] << 24) | ((buf[ofs + 1] & 0x0ff) << 16)
                | ((buf[ofs + 2] & 0x0ff) << 8) | (buf[ofs + 3] & 0x0ff);
    }

    // ////////////////////////////////////////////////////////////////
```

```
public final static void intToByteArray(int value, byte[] buf, int ofs) {
        buf[ofs] = (byte) ((value >>> 24) & 0x0ff);
        buf[ofs + 1 ] = (byte) ((value >>> 16) & 0x0ff);
        buf[ofs + 2] = (byte) ((value >>> 8) & 0x0ff);
        buf[ofs + 3] = (byte) value;
}

// ////////////////////////////////////////////////////////////////////////

public final static long byteArrayToLong(byte[] buf, int ofs) {
        // (optimized for 32bit platforms)

        return ((long) ((buf[ofs] << 24) | ((buf[ofs + 1 ]& 0x0ff) << 16)
                        | ((buf[ofs + 2] & 0x0ff) << 8) | (buf[ofs + 3] & 0x0ff)) << 32)
                        | ((long) ((buf[ofs + 4 ]<< 24)
                                        | ((buf[ofs + 5] & 0x0ff) << 16)
                                        | ((buf[ofs + 6] & 0x0ff) << 8) | (buf[ofs + 7] & 0x0ff)) & 0x0ffffffffL);
}

// ////////////////////////////////////////////////////////////////////////

public final static void longToByteArray(long value, byte[] buf, int ofs) {
        int tmp = (int) (value >>> 32);

        buf[ofs] = (byte) (tmp >>> 24);
        buf[ofs + 1 ] = (byte) ((tmp >>> 16) & 0x0ff);
        buf[ofs + 2 ] = (byte) ((tmp >>> 8) & 0x0ff);
        buf[ofs + 3] = (byte) tmp;

        tmp = (int) value;

        buf[ofs + 4 ]= (byte) (tmp >>> 24);
        buf[ofs + 5] = (byte) ((tmp >>> 16) & 0x0ff);
        buf[ofs + 6 ]= (byte) ((tmp >>> 8) & 0x0ff);
        buf[ofs + 7] = (byte) tmp;
}

// ////////////////////////////////////////////////////////////////////////

public final static long intArrayToLong(int[] buf, int ofs) {
        return (((long) buf[ofs]) << 32)
                        | (((long) buf[ofs + 1]) & 0x0ffffffffL);
}

// ////////////////////////////////////////////////////////////////////////

public final static void longToIntArray(long value, int[] buf, int ofs) {
        buf[ofs] = (int) (value >>> 32);
        buf[ofs + 1 ] = (int) value;
}

// ////////////////////////////////////////////////////////////////////////

public final static long makeLong(int lo, int hi) {
        return (((long) hi << 32) | ((long) lo & 0x00000000ffffffffL));
}

// ////////////////////////////////////////////////////////////////////////
```

```
    public final static int longLo32(long val) {
        return (int) val;
    }

    // ////////////////////////////////////////////////////////////////////

    public final static int longHi32(long val) {
        return (int) (val >>> 32);
    }

    // ////////////////////////////////////////////////////////////////////

    final static char[] HEXTAB = { '0', '1', '2', '3', '4', '5', '6', '7', '8',
            '9', 'a', 'b', 'c', 'd', 'e', 'f' };

    public final static String bytesToHexStr(byte[] data) {
        return bytesToHexStr(data, 0, data.length);
    }

    // ////////////////////////////////////////////////////////////////////

    public final static String bytesToHexStr(byte[] data, int ofs, int len) {
        StringBuffer sbuf = new StringBuffer();
        sbuf.setLength(len << 1);

        int pos = 0;
        int c = ofs + len;

        while (ofs < c) {
            sbuf.setCharAt(pos++, HEXTAB[(data[ofs] >> 4) & 0x0f]);
            sbuf.setCharAt(pos++, HEXTAB[data[ofs++] & 0x0f]);
        }
        return sbuf.toString();
    }

    // ////////////////////////////////////////////////////////////////////

    public final static int hexStrToBytes(String hex, byte[] data, int srcofs,
            int dstofs, int len) {
        // check for correct ranges

        final int strlen = hex.length();

        int availBytes = (strlen - srcofs) >> 1;
        if (availBytes < len) {
            len = availBytes;
        }

        final int outputCapacity = data.length - dstofs;
        if (len > outputCapacity) {
            len = outputCapacity;
        }

        // convert now

        final int dstofsBak = dstofs;
```

```
        for (int i= 0; i < len; i++) {
            byte abyte = 0;
            boolean convertOK = true;

            for (int j = 0; j<2; j++) {
                abyte <<= 4;
                char cActChar = hex.charAt(srcofs++);

                if ((cActChar >= 'a') && (cActChar <= 'f')) {
                    abyte |= (byte) (cActChar - 'a') + 10;
                } else {
                    if ((cActChar >= '0') && (cActChar <= '9')) {
                        abyte |= (byte) (cActChar - '0');
                    } else {
                        convertOK = false;
                    }
                }
            }
            if (convertOK) {
                data[dstofs++]= abyte;
            }
        }

        return (dstofs - dstofsBak);
    }

    // ////////////////////////////////////////////////////////////////////

    public final static String byteArrayToStr(byte[] data, int ofs, int len) {
        // we need two bytes for every character
        len &= ~1;

        // enough bytes in the buffer?
        final int availCapacity = data.length - ofs;

        if (availCapacity < len) {
            len = availCapacity;
        }

        final StringBuffer sbuf = new StringBuffer();
        sbuf.setLength(len >> 1);

        int sbufPos = 0;

        while (0 < len) {
            sbuf.setCharAt(sbufPos++,
                    (char) ((data[ofs] << 8) | (data[ofs + 1] & 0x0ff)));
            ofs += 2;
            len -= 2;
        }

        return sbuf.toString();
    }
```

```
package my.project.scop;

public class Scop {
	public final static int MAXKEYLENGTH = 56;

	public final static int BLOCKSIZE = 8;
}
```

ScopCBC.java

```
package my.project.scop;

public final class ScopCBC extends ScopECB {
	// the initialization vector (IV)
	int ivLo;

	int ivHi;

	// ////////////////////////////////////////////////////////////////////

	public long getCBCIV() {
		return BinConverter.makeLong(ivLo, ivHi);
	}

	// ////////////////////////////////////////////////////////////////////

	public void getCBCIV(byte[] dest, int ofs) {
		BinConverter.intToByteArray(ivHi, dest, ofs);
		BinConverter.intToByteArray(ivLo, dest, ofs + 4);
	}

	// ////////////////////////////////////////////////////////////////////

	public void setCBCIV(long newIV) {
		ivHi = BinConverter.longHi32(newIV);
		ivLo = BinConverter.longLo32(newIV);
	}

	// ////////////////////////////////////////////////////////////////////

	public void setCBCIV(byte[] newIV, int ofs) {
		ivHi = BinConverter.byteArrayToInt(newIV, ofs);
		ivLo = BinConverter.byteArrayToInt(newIV, ofs + 4);
	}

	// ////////////////////////////////////////////////////////////////////

	public ScopCBC(byte[] key, int ofs, int len) {
		super(key, ofs, len);
	}

	// ////////////////////////////////////////////////////////////////////

	public ScopCBC(byte[] key, int ofs, int len, long initIV) {
		super(key, ofs, len);
		setCBCIV(initIV);
```

```
    }

    // ////////////////////////////////////////////////////////////////

    public ScopCBC(byte[] key, int ofs, int len, byte[] initIV, int ivOfs) {
        super(key, ofs, len);
        setCBCIV(initIV, ivOfs);
    }

    // ////////////////////////////////////////////////////////////////

    public void cleanUp() {
        ivHi = ivLo = 0;
        super.cleanUp();
    }

    // ////////////////////////////////////////////////////////////////

    public int encrypt(byte[] inbuf, int inpos, byte[] outbuf, int outpos,
            int len) {
        len -= len % BLOCKSIZE;

        final int c = inpos + len;

        final int[] pbox = this.pbox;
        final int pbox00 = pbox[0];
        final int pbox01 = pbox[1];
        final int pbox02 = pbox[2];
        final int pbox03 = pbox[3];
        final int pbox04 = pbox[4];
        final int pbox05 = pbox[5];
        final int pbox06 = pbox[6];
        final int pbox07 = pbox[7];
        final int pbox08 = pbox[8];
        final int pbox09 = pbox[9];
        final int pbox10 = pbox[10];
        final int pbox11 = pbox[11];
        final int pbox12 = pbox[12];
        final int pbox13 = pbox[13];
        final int pbox14 = pbox[14];
        final int pbox15 = pbox[15];
        final int pbox16 = pbox[16];
        final int pbox17 = pbox[17];

        final int[] sbox1 = this.sbox1;
        final int[] sbox2 = this.sbox2;
        final int[] sbox3 = this.sbox3;
        final int[] sbox4 = this.sbox4;

        int ivHi = this.ivHi;
        int ivLo = this.ivLo;

        int hi, lo;

        while (inpos < c) {
            hi = (inbuf[inpos] << 24) | ((inbuf[inpos + 1] << 16) & 0x0ff0000)
                    | ((inbuf[inpos + 2] << 8) & 0x000ff00)
                    | (inbuf[inpos + 3] & 0x00000ff);
```

```
                lo = (inbuf[inpos + 4] << 24)
                        | ((inbuf[inpos + 5] << 16) & 0x0ff0000)
                        | ((inbuf[inpos + 6] << 8) & 0x000ff00)
                        | (inbuf[inpos + 7] & 0x00000ff);

                inpos += 8;

                hi ^= ivHi;
                lo^= ivLo;

                hi ^= pbox00;
                lo ^= (((sbox1[hi >>> 24] + sbox2[(hi >>> 16) & 0x0ff]) ^ sbox3[(hi >>> 8) & 0x0ff]) +
[hi & 0x0ff]) ^ pbox01;
                hi^= (((sbox1[lo >>> 24] + sbox2[(lo >>> 16) & 0x0ff]) ^ sbox3[(lo >>> 8) & 0x0ff]) +
[lo & 0x0ff]) ^ pbox02;
                lo^=(((sbox1[hi >>> 24] + sbox2[(hi >>> 16) & 0x0ff]) ^ sbox3[(hi >>> 8) & 0x0ff]) +
[hi & 0x0ff]) ^ pbox03;
                hi^= (((sbox1[lo >>> 24] + sbox2[(lo >>> 16) & 0x0ff]) ^ sbox3[(lo >>> 8) & 0x0ff]) +
[lo & 0x0ff]) ^ pbox04;
                lo ^= (((sbox1[hi >>> 24] + sbox2[(hi >>> 16) & 0x0ff]) ^ sbox3[(hi >>> 8) & 0x0ff]) +
[hi & 0x0ff]) ^ pbox05;
                hi ^= (((sbox1[lo >>> 24] + sbox2[(lo >>> 16) & 0x0ff]) ^ sbox3[(lo >>> 8) & 0x0ff]) +
[lo & 0x0ff]) ^ pbox06;
                lo ^= (((sbox1[hi >>> 24] + sbox2[(hi >>> 16) & 0x0ff]) ^ sbox3[(hi >>> 8) & 0x0ff]) +
[hi & 0x0ff]) ^ pbox07;
                hi^= (((sbox1[lo >>> 24] + sbox2[(lo >>> 16) & 0x0ff]) ^ sbox3[(lo >>> 8) & 0x0ff]) +
[lo & 0x0ff]) ^ pbox08;
                lo ^= (((sbox1[hi >>> 24] + sbox2[(hi >>> 16) & 0x0ff]) ^ sbox3[(hi >>> 8) & 0x0ff]) +
[hi & 0x0ff]) ^ pbox09;
                hi^= (((sbox1[lo >>> 24] + sbox2[(lo >>> 16) & 0x0ff]) ^ sbox3[(lo >>> 8) & 0x0ff]) +
[lo & 0x0ff]) ^ pbox10;
                lo ^= (((sbox1[hi >>> 24] + sbox2[(hi >>> 16) & 0x0ff]) ^ sbox3[(hi >>> 8) & 0x0ff]) +
[hi & 0x0ff]) ^ pbox11;
                hi ^= (((sbox1[lo >>> 24] + sbox2[(lo >>> 16) & 0x0ff]) ^ sbox3[(lo >>> 8) & 0x0ff]) +
[lo & 0x0ff]) ^ pbox12;
                lo ^= (((sbox1[hi >>> 24] + sbox2[(hi >>> 16) & 0x0ff]) ^ sbox3[(hi >>> 8) & 0x0ff]) +
[hi & 0x0ff]) ^ pbox13;
                hi ^= (((sbox1[lo >>> 24] + sbox2[(lo >>> 16) & 0x0ff]) ^ sbox3[(lo >>> 8) & 0x0ff]) +
[lo & 0x0ff]) ^ pbox14;
                lo ^= (((sbox1[hi >>> 24] + sbox2[(hi >>> 16) & 0x0ff]) ^ sbox3[(hi >>> 8) & 0x0ff]) +
[hi & 0x0ff]) ^ pbox15;
                hi ^= (((sbox1[lo >>> 24] + sbox2[(lo >>> 16) & 0x0ff]) ^ sbox3[(lo >>> 8) & 0x0ff]) +
[lo & 0x0ff]) ^ pbox16;
                lo ^= pbox17;

                outbuf[outpos] = (byte) (lo >>> 24);
                outbuf[outpos + 1] = (byte) (lo >>> 16);
                outbuf[outpos + 2] = (byte) (lo >>> 8);
                outbuf[outpos + 3] = (byte) lo;

                outbuf[outpos + 4] = (byte) (hi >>> 24);
                outbuf[outpos + 5] = (byte) (hi >>> 16);
                outbuf[outpos + 6] = (byte) (hi >>> 8);
                outbuf[outpos + 7] = (byte) hi;

                outpos += 8;
```

```
            ivHi = lo;
            ivLo= hi;
        }

        this.ivHi = ivHi;
        this.ivLo= ivLo;

        return len;
    }

    // ////////////////////////////////////////////////////////////////////////

    public int decrypt(byte[] inbuf, int inpos, byte[] outbuf, int outpos,
            int len) {
        len -= len % BLOCKSIZE;

        final int c = inpos + len;

        final int[] pbox = this.pbox;
        final int pbox00 = pbox[0];
        final int pbox01 = pbox[1];
        final int pbox02 = pbox[2];
        final int pbox03 = pbox[3];
        final int pbox04 = pbox[4];
        final int pbox05 = pbox[5];
        final int pbox06 = pbox[6];
        final int pbox07 = pbox[7];
        final int pbox08 = pbox[8];
        final int pbox09 = pbox[9];
        final int pbox10 = pbox[10];
        final int pbox11 = pbox[11];
        final int pbox12 = pbox[12];
        final int pbox13 = pbox[13];
        final int pbox14 = pbox[14];
        final int pbox15 = pbox[15];
        final int pbox16= pbox[16];
        final int pbox17 = pbox[17];

        final int[] sbox1 = this.sbox1;
        final int[] sbox2 = this.sbox2;
        final int[] sbox3 = this.sbox3;
        final int[] sbox4 = this.sbox4;

        int ivHi = this.ivHi;
        int ivLo = this.ivLo;

        int tmpHi, tmpLo;

        int hi, lo;

        while (inpos < c) {
            hi = (inbuf[inpos++] << 24) | ((inbuf[inpos++] << 16) & 0x0ff0000)
                    | ((inbuf[inpos++ ]<< 8) & 0x000ff00)
                    | (inbuf[inpos++] & 0x00000ff);

            lo = (inbuf[inpos++] << 24) | ((inbuf[inpos++] << 16) & 0x0ff0000)
                    | ((inbuf[inpos++] << 8) & 0x000ff00)
                    | (inbuf[inpos++] & 0x00000ff);
```

```
                // save the current block, it will become the new IV
                tmpHi= hi;
                tmpLo = lo;

                hi ^= pbox17;
                lo^=(((sbox1[hi>>> 24] + sbox2[(hi >>> 16) & 0x0ff]) ^ sbox3[(hi >>> 8) & 0x0ff]) +
[hi & 0x0ff]) ^ pbox16;
                hi ^= (((sbox1[lo >>> 24] + sbox2[(lo >>> 16) & 0x0ff]) ^ sbox3[(lo >>> 8) & 0x0ff]) +
[lo & 0x0ff]) ^ pbox15;
                lo ^= (((sbox1[hi >>> 24] + sbox2[(hi >>> 16) & 0x0ff]) ^ sbox3[(hi >>> 8) & 0x0ff]) +
[hi & 0x0ff]) ^ pbox14;
                hi ^= (((sbox1[lo >>> 24] + sbox2[(lo >>> 16) & 0x0ff]) ^ sbox3[(lo >>> 8) & 0x0ff]) +
[lo & 0x0ff]) ^ pbox13;
                lo ^= (((sbox1[hi >>> 24] + sbox2[(hi >>> 16) & 0x0ff]) ^ sbox3[(hi >>> 8) & 0x0ff]) +
[hi & 0x0ff]) ^ pbox12;
                hi ^= (((sbox1[lo >>> 24] + sbox2[(lo >>> 16) & 0x0ff]) ^ sbox3[(lo >>> 8) & 0x0ff]) +
[lo & 0x0ff]) ^ pbox11;
                lo^=(((sbox1[hi >>> 24] + sbox2[(hi >>> 16) & 0x0ff]) ^ sbox3[(hi >>> 8) & 0x0ff]) +
[hi & 0x0ff]) ^ pbox10;
                hi ^= (((sbox1[lo >>> 24] + sbox2[(lo >>> 16) & 0x0ff]) ^ sbox3[(lo >>> 8) & 0x0ff]) +
[lo & 0x0ff]) ^ pbox09;
                lo ^= (((sbox1[hi >>> 24] + sbox2[(hi >>> 16) & 0x0ff]) ^ sbox3[(hi >>> 8) & 0x0ff]) +
[hi & 0x0ff]) ^ pbox08;
                hi ^= (((sbox1[lo >>> 24] + sbox2[(lo >>> 16) & 0x0ff]) ^ sbox3[(lo >>> 8) & 0x0ff]) +
[lo & 0x0ff]) ^ pbox07;
                lo ^= (((sbox1[hi >>> 24] + sbox2[(hi >>> 16) & 0x0ff]) ^ sbox3[(hi >>> 8) & 0x0ff]) +
[hi & 0x0ff]) ^ pbox06;
                hi ^= (((sbox1[lo >>> 24] + sbox2[(lo >>> 16) & 0x0ff]) ^ sbox3[(lo >>> 8) & 0x0ff]) +
[lo & 0x0ff]) ^ pbox05;
                lo ^= (((sbox1[hi >>> 24] + sbox2[(hi >>> 16) & 0x0ff]) ^ sbox3[(hi >>> 8) & 0x0ff]) +
[hi & 0x0ff]) ^ pbox04;
                hi ^= (((sbox1[lo >>> 24] + sbox2[(lo >>> 16) & 0x0ff]) ^ sbox3[(lo >>> 8) & 0x0ff]) +
[lo & 0x0ff]) ^ pbox03;
                lo ^= (((sbox1[hi >>> 24] + sbox2[(hi >>> 16) & 0x0ff]) ^ sbox3[(hi >>> 8) & 0x0ff]) +
[hi & 0x0ff]) ^ pbox02;
                hi ^= (((sbox1[lo >>> 24] + sbox2[(lo >>> 16) & 0x0ff]) ^ sbox3[(lo >>> 8) & 0x0ff]) +
[lo & 0x0ff]) ^ pbox01;
                lo ^= pbox00;

                hi ^= ivLo;
                lo ^= ivHi;

                outbuf[outpos++] = (byte) (lo >>> 24);
                outbuf[outpos++] = (byte) (lo >>> 16);
                outbuf[outpos++] = (byte) (lo >>> 8);
                outbuf[outpos++] = (byte) lo;

                outbuf[outpos++]= (byte) (hi >>> 24);
                outbuf[outpos++] = (byte) (hi >>> 16);
                outbuf[outpos++] = (byte) (hi >>> 8);
                outbuf[outpos++] = (byte) hi;

                ivHi = tmpHi;
                ivLo = tmpLo;
        }

        this.ivHi = ivHi;
```

```
            this.ivLo = ivLo;

            return len;
        }
}
```

ScopECB.java

```
package my.project.scop;

public class ScopECB extends Scop {
    // size of the single boxes
    final static int PBOX_ENTRIES = 18;

    final static int SBOX_ENTRIES = 256;

    // ////////////////////////////////////////////////////////////////////

    // The boxes. Although cloning the static ones we do it differently to make
    // the code run on the J2ME platform.
    final int[] pbox = new int[PBOX_ENTRIES];

    final int[] sbox1 = new int[SBOX_ENTRIES];

    final int[] sbox2 = new int[SBOX_ENTRIES];

    final int[] sbox3 = new int[SBOX_ENTRIES];

    final int[] sbox4 = new int[SBOX_ENTRIES];

    // The local block cache to speed up integer and long buffer handling.
    final byte[] blockBuf = new byte[BLOCKSIZE];

    // The weak key indicator. Determined on demand. -1 stands for "not known
    // yet", 0 for "no weak key" and 1 for "weak key detected".
    int weakKey = -1;

    // ////////////////////////////////////////////////////////////////////

    public void initialize(byte[] key, int ofs, int len) {
        System.arraycopy(PBOX_INIT, 0, this.pbox, 0, 18);
        System.arraycopy(SBOX1_INIT, 0, this.sbox1, 0, 256);
        System.arraycopy(SBOX2_INIT, 0, this.sbox2, 0, 256);
        System.arraycopy(SBOX3_INIT, 0, this.sbox3, 0, 256);
        System.arraycopy(SBOX4_INIT, 0, this.sbox4, 0, 256);

        if (0 == len) {
            return;
        }

        final int ofsBak = ofs;
        final int end = ofs + len;

        final int[] pbox = this.pbox;
        final int[] sbox1 = this.sbox1;
        final int[] sbox2 = this.sbox2;
```

```
        final int[] sbox3 = this.sbox3;
        final int[] sbox4 = this.sbox4;
        final byte[] blockBuf = this.blockBuf;

        // XOR the key over the p-boxes
        int build = 0;

        for (int i = 0; i < PBOX_ENTRIES; i++) {
            for (int j= 0; j<4;j++) {
                build = (build << 8) | (((int) key[ofs]) & 0x0ff);

                if (++ofs == end) {
                    ofs = ofsBak;
                }
            }
            pbox[i] ^= build;
        }

        // encrypt all boxes with the all zero string

        java.util.Arrays.fill(blockBuf, (byte) 0);

        for (int i = 0; i < PBOX_ENTRIES; i+= 2) {
            encryptPrv(blockBuf, 0, blockBuf, 0, BLOCKSIZE);
            pbox[i] = BinConverter.byteArrayToInt(blockBuf, 0);
            pbox[i + 1] = BinConverter.byteArrayToInt(blockBuf, 4);
        }

        for (int i = 0; i < SBOX_ENTRIES; i+= 2) {
            encryptPrv(blockBuf, 0, blockBuf, 0, BLOCKSIZE);
            sbox1[i] = BinConverter.byteArrayToInt(blockBuf, 0);
            sbox1[i + 1] = BinConverter.byteArrayToInt(blockBuf, 4);
        }

        for (int i = 0; i < SBOX_ENTRIES; i +=2) {
            encryptPrv(blockBuf, 0, blockBuf, 0, BLOCKSIZE);
            sbox2[i] = BinConverter.byteArrayToInt(blockBuf, 0);
            sbox2[i + 1] = BinConverter.byteArrayToInt(blockBuf, 4);
        }

        for (int i= 0; i < SBOX_ENTRIES; i += 2) {
            encryptPrv(blockBuf, 0, blockBuf, 0, BLOCKSIZE);
            sbox3[i] = BinConverter.byteArrayToInt(blockBuf, 0);
            sbox3[i + 1] = BinConverter.byteArrayToInt(blockBuf, 4);
        }

        for (int i= 0; i < SBOX_ENTRIES; i += 2) {
            encryptPrv(blockBuf, 0, blockBuf, 0, BLOCKSIZE);
            sbox4[i] = BinConverter.byteArrayToInt(blockBuf, 0);
            sbox4[i + 1] = BinConverter.byteArrayToInt(blockBuf, 4);
        }
    }

    // ////////////////////////////////////////////////////////////////////////

    public ScopECB() {
    }
```

```
// ////////////////////////////////////////////////////////////////////

public ScopECB (byte[] key, int ofs, int len) {
    initialize(key, ofs, len);
}

// ////////////////////////////////////////////////////////////////////

public void cleanUp() {
    java.util.Arrays.fill(this.pbox, 0);
    java.util.Arrays.fill(this.sbox1, 0);
    java.util.Arrays.fill(this.sbox2, 0);
    java.util.Arrays.fill(this.sbox3, 0);
    java.util.Arrays.fill(this.sbox4, 0);

    this.weakKey = -1;
}

// ////////////////////////////////////////////////////////////////////

// test vector #1 (checking for the "signed bug")
final static byte[] TESTKEY1 = { (byte) 0x1c, (byte) 0x58, (byte) 0x7f,
        (byte) 0x1c, (byte) 0x13, (byte) 0x92, (byte) 0x4f, (byte) 0xef };

final static long tvP1 = 0x305532286d6f295aL;

final static long tvC1 = 0x55cb3774d13ef201L;

// test vector #2 (official vector by Bruce Schneier)
final static String TESTKEY2 = "Who is John Galt?";

final static long tvP2 = 0xfiedcba9876543210L;

final static long tvC2 = 0xcc91732b8022f684L;

public static boolean selfTest() {
    final byte[] tvP1b = new byte[8];
    final byte[] tvT1b = new byte[8];
    final byte[] tvP2b = new byte[8];
    final byte[] tvT2b = new byte[8];
    ScopECB testbf1;
    ScopECB testbf2;

    // start the tests, check for a proper decryption, too

    testbf1 = new ScopECB(TESTKEY1, 0, TESTKEY1.length);

    BinConverter.longToByteArray(tvP1, tvP1b, 0);
    testbf1.encrypt(tvP1b, 0, tvT1b, 0, tvP1b.length);

    if (tvC1 != BinConverter.byteArrayToLong(tvT1b, 0)) {
        return false;
    }

    testbf1.decrypt(tvT1b, 0, tvT1b, 0, tvT1b.length);

    if (tvP1 != BinConverter.byteArrayToLong(tvT1b, 0)) {
        return false;
```

```
        }

        testbf2 = new ScopECB(TESTKEY2.getBytes(), 0, TESTKEY2.length());

        BinConverter.longToByteArray(tvP2, tvP2b, 0);
        testbf2.encrypt(tvP2b, 0, tvT2b, 0, tvP2b.length);

        if (tvC2 != BinConverter.byteArrayToLong(tvT2b, 0)) {
            return false;
        }

        testbf2.decrypt(tvT2b, 0, tvT2b, 0, tvT2b.length);

        if (tvP2 != BinConverter.byteArrayToLong(tvT2b, 0)) {
            return false;
        }

        return true;
    }

    // ////////////////////////////////////////////////////////////////

    public boolean weakKeyCheck() {
        if (-1 != this.weakKey) {
            return (1 == this.weakKey);
        }

        // a weak key is defined to create identical entries in at least one of
        // the s-boxes...

        int[] sbox1 = this.sbox1;
        int[] sbox2 = this.sbox2;
        int[] sbox3 = this.sbox3;
        int[] sbox4 = this.sbox4;

        for (int i = 0; i < SBOX_ENTRIES - 1; i++) {
            for (int j = i + 1; j < SBOX_ENTRIES; j++) {
                if ((sbox1[i] == sbox1[j]) | (sbox2[i] == sbox2[j])
                        | (sbox3[i] == sbox3[j]) | (sbox4[i] == sbox4[j])) {
                    this.weakKey = 1;
                    return true;
                }
            }
        }

        this.weakKey = 0;

        return false;
    }

    // ////////////////////////////////////////////////////////////////

    // Because of Java's inheritance rules we need to have this internal method
    // to avoid the Scop CBC/CFB classes to interfere.

    protected final int encryptPrv(byte[] inbuf, int inpos, byte[] outbuf,
            int outpos, int len) {
        len -= len % BLOCKSIZE;
```

```
        final int c = inpos + len;

        final int[] pbox = this.pbox;
        final int pbox00 = pbox[0];
        final int pbox01 = pbox[1];
        final int pbox02 = pbox[2];
        final int pbox03 = pbox[3];
        final int pbox04 = pbox[4];
        final int pbox05 = pbox[5];
        final int pbox06 = pbox[6];
        final int pbox07 = pbox[7];
        final int pbox08 = pbox[8];
        final int pbox09 = pbox[9];
        final int pbox10= pbox[10];
        final int pbox11 = pbox[11];
        final int pbox12 = pbox[12];
        final int pbox13 = pbox[13];
        final int pbox14 = pbox[14];
        final int pbox15 = pbox[15];
        final int pbox16 = pbox[16];
        final int pbox17 = pbox[17];

        final int[] sbox1 = this.sbox1;
        final int[] sbox2 = this.sbox2;
        final int[] sbox3 = this.sbox3;
        final int[] sbox4 = this.sbox4;

        int hi, lo;

        while (inpos < c) {
            hi = (inbuf[inpos] << 24) | ((inbuf[inpos + 1 ]<< 16) & 0x0ff0000)
                    | ((inbuf[inpos + 2 ] << 8) & 0x000ff00)
                    | (inbuf[inpos + 3] & 0x00000ff);

            lo = (inbuf[inpos + 4 ] << 24)
                    | ((inbuf[inpos + 5] << 16) & 0x0ff0000)
                    | ((inbuf[inpos + 6] << 8) & 0x000ff00)
                    | (inbuf[inpos + 7] & 0x00000ff);

            inpos += 8;

            hi ^= pbox00;
            lo ^= (((sbox1[hi >>> 24] + sbox2[(hi >>> 16) & 0x0ff]) ^ sbox3[(hi >>> 8) & 0x0ff]) +
hi & 0x0ff]) ^ pbox01;
            hi ^= (((sbox1[lo >>> 24] + sbox2[(lo >>> 16) & 0x0ff]) ^ sbox3[(lo >>> 8) & 0x0ff]) +
lo & 0x0ff]) ^ pbox02;
            lo ^= (((sbox1[hi >>> 24] + sbox2[(hi >>> 16) & 0x0ff]) ^ sbox3[(hi >>> 8) & 0x0ff]) +
hi & 0x0ff]) ^ pbox03;
            hi ^= (((sbox1[lo >>> 24] + sbox2[(lo >>> 16) & 0x0ff]) ^ sbox3[(lo >>> 8) & 0x0ff]) +
lo & 0x0ff]) ^ pbox04;
            lo ^= (((sbox1[hi >>> 24] + sbox2[(hi >>> 16) & 0x0ff]) ^ sbox3[(hi >>> 8) & 0x0ff]) +
hi & 0x0ff]) ^ pbox05;
            hi ^= (((sbox1[lo >>> 24] + sbox2[(lo >>> 16) & 0x0ff]) ^ sbox3[(lo >>> 8) & 0x0ff]) +
lo & 0x0ff]) ^ pbox06;
            lo ^= (((sbox1[hi >>> 24] + sbox2[(hi >>> 16) & 0x0ff]) ^ sbox3[(hi >>> 8) & 0x0ff]) +
hi & 0x0ff]) ^ pbox07;
            hi ^= (((sbox1[lo >>> 24] + sbox2[(lo >>> 16) & 0x0ff]) ^ sbox3[(lo >>> 8) & 0x0ff]) +
```

```
[lo & 0x0ff]) ^ pbox08;
                lo^=(((sbox1[hi >>> 24] + sbox2[(hi >>> 16) & 0x0ff]) ^ sbox3[(hi >>> 8) & 0x0ff]) +
[hi & 0x0ff]) ^ pbox09;
                hi^= (((sbox1[lo >>> 24] + sbox2[(lo >>> 16) & 0x0ff]) ^ sbox3[(lo >>> 8) & 0x0ff]) +
[lo & 0x0ff]) ^ pbox10;
                lo^=(((sbox1[hi >>> 24] + sbox2[(hi >>> 16) & 0x0ff]) ^ sbox3[(hi >>> 8) & 0x0ff]) +
[hi & 0x0ff]) ^ pbox11;
                hi^= (((sbox1[lo >>> 24] + sbox2[(lo >>> 16) & 0x0ff]) ^ sbox3[(lo >>> 8) & 0x0ff]) +
[lo & 0x0ff]) ^ pbox12;
                lo^=(((sbox1[hi >>> 24] + sbox2[(hi >>> 16) & 0x0ff]) ^ sbox3[(hi >>> 8) & 0x0ff]) +
[hi & 0x0ff]) ^ pbox13;
                hi^= (((sbox1[lo >>> 24] + sbox2[(lo >>> 16) & 0x0ff]) ^ sbox3[(lo >>> 8) & 0x0ff]) +
[lo & 0x0ff]) ^ pbox14;
                lo^=(((sbox1[hi>>> 24] + sbox2[(hi >>> 16) & 0x0ff]) ^ sbox3[(hi >>> 8) & 0x0ff]) +
[hi & 0x0ff]) ^ pbox15;
                hi^= (((sbox1[lo >>> 24] + sbox2[(lo >>> 16) & 0x0ff]) ^ sbox3[(lo >>> 8) & 0x0ff]) +
[lo & 0x0ff]) ^ pbox16;
                lo ^= pbox17;

                outbuf[outpos] = (byte) (lo >>> 24);
                outbuf[outpos + 1 ]= (byte) (lo >>> 16);
                outbuf[outpos + 2 ]= (byte) (lo >>> 8);
                outbuf[outpos + 3 ]= (byte) lo;

                outbuf[outpos + 4 ]= (byte) (hi >>> 24);
                outbuf[outpos + 5] = (byte) (hi>>> 16);
                outbuf[outpos + 6] = (byte) (hi >>> 8);
                outbuf[outpos + 7] = (byte) hi;

                outpos += 8;
        }

        return len;
    }

    // ////////////////////////////////////////////////////////////////

    public int encrypt(byte[] inbuf, int inpos, byte[] outbuf, int outpos,
            int len) {
        return encryptPrv(inbuf, inpos, outbuf, outpos, len);
    }

    // ////////////////////////////////////////////////////////////////

    public int decrypt(byte[] inbuf, int inpos, byte[] outbuf, int outpos,
            int len) {
        len -= len % BLOCKSIZE;

        final int c = inpos + len;

        final int[] pbox = this.pbox;
        final int pbox00 = pbox[0];
        final int pbox01 = pbox[1];
        final int pbox02 = pbox[2];
        final int pbox03 = pbox[3];
        final int pbox04 = pbox[4];
        final int pbox05 = pbox[5];
        final int pbox06 = pbox[6];
```

```
        final int pbox07 = pbox[7];
        final int pbox08 = pbox[8];
        final int pbox09 = pbox[9];
        final int pbox10 = pbox[10];
        final int pbox11 = pbox[11];
        final int pbox12= pbox[12];
        final int pbox13 = pbox[13];
        final int pbox14 = pbox[14];
        final int pbox15 = pbox[15];
        final int pbox16 = pbox[16];
        final int pbox17 = pbox[17];

        final int[] sbox1 = this.sbox1;
        final int[] sbox2 = this.sbox2;
        final int[] sbox3 = this.sbox3;
        final int[] sbox4 = this.sbox4;

        int hi, lo;

        while (inpos < c) {
            hi = (inbuf[inpos] << 24) | ((inbuf[inpos + 1] << 16) & 0x0ff0000)
                    | ((inbuf[inpos + 2] << 8) & 0x000ff00)
                    | (inbuf[inpos + 3] & 0x00000ff);

            lo = (inbuf[inpos + 4] << 24)
                    | ((inbuf[inpos + 5] << 16) & 0x0ff0000)
                    | ((inbuf[inpos + 6] << 8) & 0x000ff00)
                    | (inbuf[inpos + 7] & 0x00000ff);

            inpos += 8;

            hi ^= pbox17;
            lo ^= (((sbox1[hi >>> 24] + sbox2[(hi >>> 16) & 0x0ff]) ^ sbox3[(hi >>> 8) & 0x0ff]) +
hi & 0x0ff]) ^ pbox16;
            hi ^= (((sbox1[lo >>> 24] + sbox2[(lo >>> 16) & 0x0ff]) ^ sbox3[(lo >>> 8) & 0x0ff]) +
lo & 0x0ff]) ^ pbox15;
            lo ^= (((sbox1[hi >>> 24] + sbox2[(hi >>> 16) & 0x0ff]) ^ sbox3[(hi >>> 8) & 0x0ff]) +
hi & 0x0ff]) ^ pbox14;
            hi ^= (((sbox1[lo >>> 24] + sbox2[(lo >>> 16) & 0x0ff]) ^ sbox3[(lo >>> 8) & 0x0ff]) +
lo & 0x0ff]) ^ pbox13;
            lo ^= (((sbox1[hi >>> 24] + sbox2[(hi >>> 16) & 0x0ff]) ^ sbox3[(hi >>> 8) & 0x0ff]) +
hi & 0x0ff]) ^ pbox12;
            hi ^= (((sbox1[lo >>> 24] + sbox2[(lo >>> 16) & 0x0ff]) ^ sbox3[(lo >>> 8) & 0x0ff]) +
lo & 0x0ff]) ^ pbox11;
            lo ^= (((sbox1[hi >>> 24] + sbox2[(hi >>> 16) & 0x0ff]) ^ sbox3[(hi >>> 8) & 0x0ff]) +
hi & 0x0ff]) ^ pbox10;
            hi ^= (((sbox1[lo >>> 24] + sbox2[(lo >>> 16) & 0x0ff]) ^ sbox3[(lo >>> 8) & 0x0ff]) +
lo & 0x0ff]) ^ pbox09;
            lo ^= (((sbox1[hi >>> 24] + sbox2[(hi >>> 16) & 0x0ff]) ^ sbox3[(hi >>> 8) & 0x0ff]) +
hi & 0x0ff]) ^ pbox08;
            hi ^= (((sbox1[lo >>> 24] + sbox2[(lo >>> 16) & 0x0ff]) ^ sbox3[(lo >>> 8) & 0x0ff]) +
lo & 0x0ff]) ^ pbox07;
            lo ^= (((sbox1[hi >>> 24] + sbox2[(hi >>> 16) & 0x0ff]) ^ sbox3[(hi >>> 8) & 0x0ff]) +
hi & 0x0ff]) ^ pbox06;
            hi ^= (((sbox1[lo >>> 24] + sbox2[(lo >>> 16) & 0x0ff]) ^ sbox3[(lo >>> 8) & 0x0ff]) +
lo & 0x0ff]) ^ pbox05;
            lo ^= (((sbox1[hi >>> 24] + sbox2[(hi >>> 16) & 0x0ff]) ^ sbox3[(hi >>> 8) & 0x0ff]) +
hi & 0x0ff]) ^ pbox04;
```

```
                hi ^= (((sbox1[lo >>> 24] + sbox2[(lo >>> 16) & 0x0ff]) ^ sbox3[(lo >>> 8) & 0x0ff]) +
[lo & 0x0ff]) ^ pbox03;
                lo ^= (((sbox1[hi >>> 24] + sbox2[(hi >>> 16) & 0x0ff]) ^ sbox3[(hi >>> 8) & 0x0ff]) +
[hi & 0x0ff]) ^ pbox02;
                hi ^= (((sbox1[lo >>> 24] + sbox2[(lo >>> 16) & 0x0ff]) ^ sbox3[(lo >>> 8) & 0x0ff]) +
[lo & 0x0ff]) ^ pbox01;
                lo ^= pbox00;

                outbuf[outpos] = (byte) (lo >>> 24);
                outbuf[outpos + 1 ] = (byte) (lo >>> 16);
                outbuf[outpos + 2 ] = (byte) (lo >>> 8);
                outbuf[outpos + 3 ] = (byte) lo;

                outbuf[outpos + 4 ] = (byte) (hi >>> 24);
                outbuf[outpos + 5] = (byte) (hi >>> 16);
                outbuf[outpos + 6] = (byte) (hi >>> 8);
                outbuf[outpos + 7] = (byte) hi;

                outpos += 8;
        }

        return len;
    }

    // ////////////////////////////////////////////////////////////////

    // Initialization data for all the boxes.

    final static int PBOX_INIT[ ]= {0x243f6a88, 0x85a308d3, 0x13198a2e,
                0x03707344, 0xa4093822, 0x299f31d0, 0x082efa98, 0xec4e6c89,
                0x452821e6, 0x38d01377, 0xbe5466cf, 0x34e90c6c, 0xc0ac29b7,
                0xc97c50dd, 0x3f84d5b5, 0xb5470917, 0x9216d5d9, 0x8979fb1b};

    final static int SBOX1_INIT[] = { 0xd1310ba6, 0x98dfb5ac, 0x2ffd72db,
                0xd01adfb7, 0xb8e1afed, 0x6a267e96, 0xba7c9045, 0xf12c7f99,
                0x24a19947, 0xb3916cf7, 0x0801f2e2,0x858efc16, 0x636920d8,
                0x71574e69, 0xa458fea3, 0xf4933d7e, 0x0d95748f, 0x728eb658,
                0x718bcd58, 0x82154aee, 0x7b54a41d,0xc25a59b5,0x9c30d539,
                0x2af26013, 0xc5d1b023, 0x286085f0, 0xca417918, 0xb8db38ef,
                0x8e79dcb0,0x603a180e, 0x6c9e0e8b,0xb01e8a3e,0xd71577c1,
                0xbd314b27,0x78af2fda,0x55605c60,0xe65525f3,0xaa55ab94,
                0x57489862, 0x63e81440, 0x55ca396a, 0x2aab10b6, 0xb4cc5c34,
                0x1141e8ce, 0xa15486af, 0x7c72e993, 0xb3ee1411, 0x636fbc2a,
                0x2ba9c55d,0x741831f6,0xce5c3e16,0x9b87931e, 0xafd6ba33,
                0x6c24cf5c, 0x7a325381, 0x28958677, 0x3b8f4898, 0x6b4bb9af,
                0xc4bfe81b,0x66282193,0x61d809cc,0xfb21a991,0x487cac60,
                0x5dec8032,0xef845d5d, 0xe98575b1,0xdc262302, 0xeb651b88,
                0x23893e81, 0xd396acc5, 0x0f6d6ff3, 0x83f44239, 0x2e0b4482,
                0xa4842004, 0x69c8f04a, 0x9e1f9b5e, 0x21c66842, 0xf6e96c9a,
                0x670c9c61, 0xabd388f0, 0x6a51a0d2, 0xd8542f68, 0x960fa728,
                0xab5133a3, 0x6eef0b6c, 0x137a3be4,0xba3bf050,0x7efb2a98,
                0xa1f1651d, 0x39af0176, 0x66ca593e, 0x82430e88, 0x8cee8619,
                0x456f9fb4, 0x7d84a5c3, 0x3b8b5ebe, 0xe06f75d8, 0x85c12073,
                0x401a449f, 0x56c16aa6, 0x4ed3aa62, 0x363f7706, 0x1bfedf72,
                0x429b023d, 0x37d0d724, 0xd00a1248, 0xdb0fead3, 0x49f1c09b,
                0x075372c9,0x80991b7b, 0x25d479d8,0xf6e8def7,0xe3fe501a,
                0xb6794c3b, 0x976ce0bd, 0x04c006ba, 0xc1a94fb6, 0x409f60c4,
                0x5e5c9ec2, 0x196a2463, 0x68fb6faf, 0x3e6c53b5, 0x1339b2eb,
```

```
            0x3b52ec6f, 0x6dfc511f, 0x9b30952c, 0xcc814544, 0xaf5ebd09,
            0xbee3d004, 0xde334afd,0x660f2807, 0x192e4bb3,0xc0cba857,
            0x45c8740f, 0xd20b5f39, 0xb9d3fbdb, 0x5579c0bd, 0x1a60320a,
            0xd6a100c6, 0x402c7279, 0x679f25fe, 0xfb1fa3cc, 0x8ea5e9f8,
            0xdb3222f8, 0x3c7516df, 0xfd616b15, 0x2f501ec8, 0xad0552ab,
            0x323db5fa, 0xfd238760, 0x53317b48, 0x3e00df82, 0x9e5c57bb,
            0xca6f8ca0,0x1a87562e,0xdf1769db,0xd542a8f6,0x287effc3,
            0xac6732c6, 0x8c4f5573, 0x695b27b0, 0xbbca58c8, 0xe1ffa35d,
            0xb8f011a0, 0x10fa3d98, 0xfd2183b8, 0x4afcb56c, 0x2dd1d35b,
            0x9a53e479, 0xb6f84565, 0xd28e49bc, 0x4bfb9790, 0xe1ddf2da,
            0xa4cb7e33, 0x62fb1341,0xcee4c6e8, 0xef20cada,0x36774c01,
            0xd07e9efe, 0x2bf11fb4, 0x95dbda4d, 0xae909198, 0xeaad8e71,
            0x6b93d5a0, 0xd08ed1d0, 0xafc725e0, 0x8e3c5b2f, 0x8e7594b7,
            0x8ff6e2fb, 0xf2122b64, 0x8888b812, 0x900df01c, 0x4fad5ea0,
            0x688fc31c, 0xd1cff191, 0xb3a8c1ad, 0x2f2f2218, 0xbe0e1777,
            0xea752dfe, 0x8b021fa1, 0xe5a0cc0f, 0xb56f74e8, 0x18acf3d6,
            0xce89e299, 0xb4a84fe0, 0xfd13e0b7, 0x7cc43b81, 0xd2ada8d9,
            0x165fa266,0x80957705,0x93cc7314, 0x211a1477,0xe6ad2065,
            0x77b5fa86, 0xc75442f5, 0xfb9d35cf, 0xebcdaf0c, 0x7b3e89a0,
            0xd6411bd3,0xae1e7e49,0x00250e2d,0x2071b35e,0x226800bb,
            0x57b8e0af, 0x2464369b, 0xf009b91e, 0x5563911d, 0x59dfa6aa,
            0x78c14389, 0xd95a537f, 0x207d5ba2, 0x02e5b9c5, 0x83260376,
            0x6295cfa9,0x11c81968, 0x4e734a41,0xb3472dca, 0x7b14a94a,
            0x1b510052, 0x9a532915, 0xd60f573f, 0xbc9bc6e4, 0x2b60a476,
            0x81e67400, 0x08ba6fb5, 0x571be91f, 0xf296ec6b, 0x2a0dd915,
            0xb6636521, 0xe7b9f9b6, 0xff34052e, 0xc5855664, 0x53b02d5d,
            0xa99f8fa1, 0x08ba4799, 0x6e85076a };

final static int SBOX2_INIT[] = { 0x4b7a70e9, 0xb5b32944, 0xdb75092e,
            0xc4192623,0xad6ea6b0, 0x49a7df7d, 0x9cee60b8,0x8fedb266,
            0xecaa8c71,0x699a17ff,0x5664526c, 0xc2b19ee1,0x193602a5,
            0x75094c29, 0xa0591340, 0xe4183a3e, 0x3f54989a, 0x5b429d65,
            0x6b8fe4d6, 0x99f73fd6, 0xa1d29c07, 0xefe830f5, 0x4d2d38e6,
            0xf0255dc1, 0x4cdd2086, 0x8470eb26, 0x6382e9c6,0x021ecc5e,
            0x09686b3f, 0x3ebaefc9, 0x3c971814, 0x6b6a70a1, 0x687f3584,
            0x52a0e286, 0xb79c5305, 0xaa500737,0x3e07841c,0x7fdeae5c,
            0x8e7d44ec, 0x5716f2b8, 0xb03ada37, 0xf0500c0d, 0xf01c1f04,
            0x0200b3ff, 0xae0cf51a, 0x3cb574b2, 0x25837a58, 0xdc0921bd,
            0xd19113f9, 0x7ca92ff6, 0x94324773, 0x22f54701,0x3ae5e581,
            0x37c2dadc,0xc8b57634,0x9af3dda7,0xa9446146,0x0fd0030e,
            0xecc8c73e, 0xa4751e41, 0xe238cd99, 0x3bea0e2f, 0x3280bba1,
            0x183eb331, 0x4e548b38, 0x4f6db908, 0x6f420d03, 0xf60a04bf,
            0x2cb81290, 0x24977c79, 0x5679b072, 0xbcaf89af, 0xde9a771f,
            0xd9930810, 0xb38bae12, 0xdccf3f2e, 0x5512721f, 0x2e6b7124,
            0x501adde6,0x9f84cd87, 0x7a584718, 0x7408da17, 0xbc9f9abc,
            0xe94b7d8c,0xec7aec3a, 0xdb851dfa, 0x63094366,0xc464c3d2,
            0xef1c1847, 0x3215d908, 0xdd433b37, 0x24c2ba16, 0x12a14d43,
            0x2a65c451, 0x50940002, 0x133ae4dd, 0x71dff89e, 0x10314e55,
            0x81ac77d6,0x5f11199b, 0x043556f1, 0xd7a3c76b, 0x3c11183b,
            0x5924a509, 0xf28fe6ed, 0x97f1fbfa, 0x9ebabf2c, 0x1e153c6e,
            0x86e34570, 0xeae96fb1, 0x860e5e0a, 0x5a3e2ab3,0x771fe71c,
            0x4e3d06fa, 0x2965dcb9, 0x99e71d0f, 0x803e89d6, 0x5266c825,
            0x2e4cc978,0x9c10b36a, 0xc6150eba, 0x94e2ea78, 0xa5fc3c53,
            0x1e0a2df4, 0xf2f74ea7, 0x361d2b3d, 0x1939260f, 0x19c27960,
            0x5223a708, 0xf71312b6, 0xebadfe6e, 0xeac31f66,0xe3bc4595,
            0xa67bc883,0xb17f37d1, 0x018cff28, 0xc332ddef,0xbe6c5aa5,
            0x65582185, 0x68ab9802, 0xeecea50f, 0xdb2f953b, 0x2aef7dad,
            0x5b6e2f84, 0x1521b628, 0x29076170, 0xecdd4775,0x619f1510,
```

```
            0x13cca830, 0xeb61bd96, 0x0334fe1e, 0xaa0363cf, 0xb5735c90,
            0x4c70a239, 0xd59e9e0b, 0xcbaade14, 0xeecc86bc, 0x60622ca7,
            0x9cab5cab, 0xb2f3846e, 0x648b1eaf, 0x19bdf0ca, 0xa02369b9,
            0x655abb50,0x40685a32,0x3c2ab4b3,0x319ee9d5,0xc021b8f7,
            0x9b540b19, 0x875fa099, 0x95f7997e, 0x623d7da8, 0xf837889a,
            0x97e32d77, 0x11ed935f, 0x16681281, 0x0e358829, 0xc7e61fd6,
            0x96dedfa1, 0x7858ba99, 0x57f584a5, 0x1b227263, 0x9b83c3ff,
            0x1ac24696, 0xcdb30aeb, 0x532e3054, 0x8fd948e4, 0x6dbc3128,
            0x58ebf2ef, 0x34c6ffea, 0xfe28ed61, 0xee7c3c73, 0x5d4a14d9,
            0xe864b7e3, 0x42105d14, 0x203e13e0, 0x45eee2b6, 0xa3aaabea,
            0xdb6c4f15, 0xfacb4fd0, 0xc742f442, 0xef6abbb5, 0x654f3b1d,
            0x41cd2105, 0xd81e799e, 0x86854dc7, 0xe44b476a, 0x3d816250,
            0xcf62a1f2, 0x5b8d2646, 0xfc8883a0, 0xc1c7b6a3, 0x7f1524c3,
            0x69cb7492, 0x47848a0b, 0x5692b285, 0x095bbf00, 0xad19489d,
            0x1462b174, 0x23820e00, 0x58428d2a, 0x0c55f5ea, 0x1dadf43e,
            0x233f7061, 0x3372f092, 0x8d937e41, 0xd65fecf1, 0x6c223bdb,
            0x7cde3759, 0xcbee7460, 0x4085f2a7, 0xce77326e, 0xa6078084,
            0x19f8509e,0xe8efd855,0x61d99735,0xa969a7aa,0xc50c06c2,
            0x5a04abfc, 0x800bcadc, 0x9e447a2e, 0xc3453484, 0xfdd56705,
            0x0e1e9ec9,0xdb73dbd3, 0x105588cd, 0x675fda79, 0xe3674340,
            0xc5c43465, 0x713e38d8, 0x3d28f89e, 0xf16dff20, 0x153e21e7,
            0x8fb03d4a, 0xe6e39f2b, 0xdb83adf7};

final static int SBOX3_INIT[] = { 0xe93d5a68, 0x948140f7, 0xf64c261c,
            0x94692934, 0x411520f7, 0x7602d4f7, 0xbcf46b2e, 0xd4a20068,
            0xd4082471, 0x3320f46a, 0x43b7d4b7, 0x500061af, 0x1e39f62e,
            0x97244546, 0x14214f74, 0xbf8b8840, 0x4d95fc1d, 0x96b591af,
            0x70f4ddd3, 0x66a02f45, 0xbfbc09ec, 0x03bd9785, 0x7fac6dd0,
            0x31cb8504,0x96eb27b3, 0x55fd3941, 0xda2547e6,0xabca0a9a,
            0x28507825, 0x530429f4, 0x0a2c86da, 0xe9b66dfb, 0x68dc1462,
            0xd7486900, 0x680ec0a4, 0x27a18dee, 0x4f3ffea2, 0xe887ad8c,
            0xb58ce006, 0x7af4d6b6, 0xaace1e7c, 0xd3375fec, 0xce78a399,
            0x406b2a42, 0x20fe9e35, 0xd9f385b9, 0xee39d7ab,0x3b124e8b,
            0x1dc9faf7, 0x4b6d1856, 0x26a36631, 0xeae397b2, 0x3a6efa74,
            0xdd5b4332, 0x6841e7f7,0xca7820fb,0xfb0af54e,0xd8feb397,
            0x454056ac, 0xba489527, 0x55533a3a, 0x20838d87, 0xfe6ba9b7,
            0xd096954b, 0x55a867bc, 0xa1159a58, 0xcca92963,0x99e1db33,
            0xa62a4a56, 0x3f3125f9, 0x5ef47e1c, 0x9029317c, 0xfdf8e802,
            0x04272f70,0x80bb155c,0x05282ce3,0x95c11548,0xe4c66d22,
            0x48c1133f, 0xc70f86dc, 0x07f9c9ee, 0x41041f0f, 0x404779a4,
            0x5d886e17, 0x325f51eb, 0xd59bc0d1, 0xf2bcc18f, 0x41113564,
            0x257b7834, 0x602a9c60, 0xdff8e8a3, 0x1f636c1b, 0x0e12b4c2,
            0x02e1329e, 0xaf664fd1, 0xcad18115,0x6b2395e0, 0x333e92e1,
            0x3b240b62, 0xeebeb922, 0x85b2a20e, 0xe6ba0d99, 0xde720c8c,
            0x2da2f728, 0xd0127845, 0x95b794fd, 0x647d0862, 0xe7ccf5f0,
            0x5449a36f, 0x877d48fa, 0xc39dfd27, 0xf33e8d1e, 0x0a476341,
            0x992eff74, 0x3a6f6eab, 0xf4f8fd37, 0xa812dc60, 0xa1ebddf8,
            0x991be14c, 0xdb6e6b0d,0xc67b5510, 0x6d672c37,0x2765d43b,
            0xdcd0e804,0xf1290dc7,0xcc00ffa3,0xb5390f92,0x690fed0b,
            0x667b9ffb, 0xcedb7d9c, 0xa091cf0b, 0xd9155ea3, 0xbb132f88,
            0x515bad24, 0x7b9479bf, 0x763bd6eb, 0x37392eb3, 0xcc115979,
            0x8026e297, 0xf42e312d, 0x6842ada7, 0xc66a2b3b, 0x12754ccc,
            0x782ef11c,0x6a124237, 0xb79251e7, 0x06a1bbe6, 0x4bfb6350,
            0x1a6b1018, 0x11caedfa, 0x3d25bdd8, 0xe2e1c3c9, 0x44421659,
            0x0a121386, 0xd90cec6e, 0xd5abea2a, 0x64af674e, 0xda86a85f,
            0xbebfe988, 0x64e4c3fe, 0x9dbc8057, 0xf0f7c086, 0x60787bf8,
            0x6003604d, 0xd1fd8346, 0xf6381fb0, 0x7745ae04, 0xd736fccc,
            0x83426b33, 0xf01eab71, 0xb0804187, 0x3c005e5f, 0x77a057be,
```

```
                0xbde8ae24, 0x55464299, 0xbf582e61, 0x4e58f48f, 0xf2ddfda2,
                0xf474ef38,0x8789bdc2, 0x5366f9c3, 0xc8b38e74, 0xb475f255,
                0x46fcd9b9, 0x7aeb2661, 0x8b1ddf84, 0x846a0e79, 0x915f95e2,
                0x466e598e, 0x20b45770, 0x8cd55591, 0xc902de4c, 0xb90bace1,
                0xbb8205d0, 0x11a86248, 0x7574a99e, 0xb77f19b6, 0xe0a9dc09,
                0x662d09a1, 0xc4324633, 0xe85a1f02, 0x09f0be8c, 0x4a99a025,
                0x1d6efe10, 0x1ab93d1d, 0x0ba5a4df, 0xa186f20f, 0x2868f169,
                0xdcb7da83, 0x573906fe, 0xa1e2ce9b, 0x4fcd7f52, 0x50115e01,
                0xa70683fa, 0xa002b5c4, 0x0de6d027, 0x9af88c27, 0x773f8641,
                0xc3604c06,0x61a806b5,0xf0177a28,0xc0f586e0, 0x006058aa,
                0x30dc7d62, 0x11e69ed7, 0x2338ea63, 0x53c2dd94, 0xc2c21634,
                0xbbcbee56, 0x90bcb6de, 0xebfc7da1, 0xce591d76, 0x6f05e409,
                0x4b7c0188, 0x39720a3d, 0x7c927c24, 0x86e3725f, 0x724d9db9,
                0x1ac15bb4,0xd39eb8fc,0xed545578,0x08fca5b5,0xd83d7cd3,
                0x4dad0fc4, 0x1e50ef5e, 0xb161e6f8, 0xa28514d9, 0x6c51133c,
                0x6fd5c7e7, 0x56e14ec4, 0x362abfce, 0xddc6c837, 0xd79a3234,
                0x92638212, 0x670efa8e, 0x406000e0};

final static int SBOX4_INIT[] = { 0x3a39ce37, 0xd3faf5cf, 0xabc27737,
                0x5ac52d1b, 0x5cb0679e, 0x4fa33742, 0xd3822740, 0x99bc9bbe,
                0xd5118e9d, 0xbf0f7315, 0xd62d1c7e, 0xc700c47b, 0xb78c1b6b,
                0x21a19045, 0xb26eb1be, 0x6a366eb4, 0x5748ab2f, 0xbc946e79,
                0xc6a376d2,0x6549c2c8, 0x530ff8ee, 0x468dde7d, 0xd5730a1d,
                0x4cd04dc6, 0x2939bbdb, 0xa9ba4650, 0xac9526e8, 0xbe5ee304,
                0xa1fad5f0, 0x6a2d519a,0x63ef8ce2, 0x9a86ee22,0xc089c2b8,
                0x43242ef6, 0xa51e03aa, 0x9cf2d0a4, 0x83c061ba, 0x9be96a4d,
                0x8fe51550, 0xba645bd6, 0x2826a2f9, 0xa73a3ae1, 0x4ba99586,
                0xef5562e9, 0xc72fefd3, 0xf752f7da, 0x3f046f69, 0x77fa0a59,
                0x80e4a915,0x87b08601,0x9b09e6ad, 0x3b3ee593,0xe990fd5a,
                0x9e34d797, 0x2cf0b7d9, 0x022b8b51,0x96d5ac3a, 0x017da67d,
                0xd1cf3ed6, 0x7c7d2d28, 0x1f9f25cf, 0xadf2b89b, 0x5ad6b472,
                0x5a88f54c, 0xe029ac71, 0xe019a5e6, 0x47b0acfd, 0xed93fa9b,
                0xe8d3c48d,0x283b57cc, 0xf8d56629, 0x79132e28, 0x785f0191,
                0xed756055, 0xf7960e44, 0xe3d35e8c, 0x15056dd4, 0x88f46dba,
                0x03a16125,0x0564f0bd,0xc3eb9e15,0x3c9057a2, 0x97271aec,
                0xa93a072a, 0x1b3f6d9b, 0x1e6321f5, 0xf59c66fb, 0x26dcf319,
                0x7533d928, 0xb155fdf5, 0x03563482, 0x8aba3cbb, 0x28517711,
                0xc20ad9f8, 0xabcc5167, 0xccad925f, 0x4de81751, 0x3830dc8e,
                0x379d5862,0x9320f991, 0xea7a90c2,0xfb3e7bce,0x5121ce64,
                0x774fbe32, 0xa8b6e37e, 0xc3293d46, 0x48de5369, 0x6413e680,
                0xa2ae0810, 0xdd6db224, 0x69852dfd, 0x09072166, 0xb39a460a,
                0x6445c0dd, 0x586cdecf, 0x1c20c8ae, 0x5bbef7dd, 0x1b588d40,
                0xccd2017f, 0x6bb4e3bb, 0xdda26a7e, 0x3a59ff45, 0x3e350a44,
                0xbcb4cdd5, 0x72eacea8, 0xfa6484bb, 0x8d6612ae, 0xbf3c6f47,
                0xd29be463, 0x542f5d9e, 0xaec2771b, 0xf64e6370,0x740e0d8d,
                0xe75b1357, 0xf8721671, 0xaf537d5d, 0x4040cb08, 0x4eb4e2cc,
                0x34d2466a, 0x0115af84, 0xe1b00428, 0x95983a1d, 0x06b89fb4,
                0xce6ea048, 0x6f3f3b82, 0x3520ab82, 0x011a1d4b, 0x277227f8,
                0x611560b1,0xe7933fdc, 0xbb3a792b, 0x344525bd, 0xa08839e1,
                0x51ce794b, 0x2f32c9b7, 0xa01fbac9, 0xe01cc87e, 0xbcc7d1f6,
                0xcf0111c3, 0xa1e8aac7, 0x1a908749, 0xd44fbd9a, 0xd0dadecb,
                0xd50ada38, 0x0339c32a, 0xc6913667, 0x8df9317c, 0xe0b12b4f,
                0xf79e59b7, 0x43f5bb3a, 0xf2d519ff, 0x27d9459c, 0xbf97222c,
                0x15e6fc2a, 0x0f91fc71, 0x9b941525, 0xfae59361, 0xceb69ceb,
                0xc2a86459, 0x12baa8d1, 0xb6c1075e, 0xe3056a0c, 0x10d25065,
                0xcb03a442, 0xe0ec6e0e, 0x1698db3b, 0x4c98a0be, 0x3278e964,
                0x9f1f9532, 0xe0d392df, 0xd3a0342b, 0x8971f21e, 0x1b0a7441,
                0x4ba3348c, 0xc5be7120, 0xc37632d8, 0xdf359f8d, 0x9b992f2e,
```

```
9                0xe60b6f47, 0x0fe3f11d, 0xe54cda54, 0x1edad891, 0xce6279cf,
0                0xcd3e7e6f, 0x1618b166, 0xfd2c1d05, 0x848fd2c5, 0xf6fb2299,
1                0xf523f357,0xa6327623,0x93a83531,0x56cccd02,0xacf08162,
2                0x5a75ebb5,0x6e163697,0x88d273cc,0xde966292,0x81b949d0,
3                0x4c50901b, 0x71c65614, 0xe6c6c7bd, 0x327a140a,0x45e1d006,
4                0xc3f27b9a, 0xc9aa53fd, 0x62a80f00, 0xbb25bfe2, 0x35bdd2f6,
5                0x71126905,0xb2040222, 0xb6cbcf7c,0xcd769c2b, 0x53113ec0,
5                0x1640e3d3, 0x38abbd60, 0x2547adf0, 0xba38209c, 0xf746ce76,
7                0x77afa1c5,0x20756060, 0x85cbfe4e,0x8ae88dd8,0x7aaaf9b0,
3                0x4cf9aa7e, 0x1948c25c, 0x02fb8a8c, 0x01c36ae4, 0xd6ebe1f9,
9                0x90d4f869, 0xa65cdea0, 0x3f09252d, 0xc208e69f, 0xb74e6132,
)                0xce77e25b, 0x578fdfe3, 0x3ac372e6 };
l }
```

ScopInputStream.java

```
package my.project.scop;

import java.io.*;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

public class ScopInputStream extends InputStream {
    PushbackInputStream is;

    ScopCBC bfc;

    byte[] buf;

    int bufPos;

    int bufCount;

    // ////////////////////////////////////////////////////////////////

    void init(byte[] key, int ofs, int len, InputStream is) throws IOException {
        this.bufPos = this.bufCount = 0;

        this.is = new PushbackInputStream(new BufferedInputStream(is));

        MessageDigest md = null;
        try {
            md = MessageDigest.getInstance("SHA-1");
        } catch (NoSuchAlgorithmException nse) {
            throw new UnsupportedOperationException();
        }

        md.update(key, ofs, len);

        byte[] ckey = md.digest();
        md.reset();

        this.bfc = new ScopCBC(ckey, 0, ckey.length, 0);

        this.buf = new byte[Scop.BLOCKSIZE];
```

```
        // read the IV
        for (int i= 0, c = this.buf.length; i <c; i++) {
            int val = this.is.read();
            if(-1== val) {
                throw new IOException("truncated stream, IV is missing");
            }
            this.buf[i] = (byte) val;
        }

        this.bfc.setCBCIV(this.buf, 0);
    }

    // ////////////////////////////////////////////////////////////////////

    void fillBuffer() throws IOException {
        // fill the whole buffer

        int val;
        for (int i = 0, c = this.buf.length; i <c; i++) {
            if (-1== (val = this.is.read())) {
                throw new IOException("truncated stream, unexpected end");
            }
            this.buf [i] = (byte) val;
        }

        // decrypt the buffer
        this.bfc.decrypt(this.buf, 0, this.buf, 0, this.buf .length);

        // peek if this is the end of the stream
        val = this.is.read();
        if(-1== val) {
            // this is the last block, so we can read out how much we actually
            // got left

            int c = this.buf[this.buf.length - 1];

            // validate the padding
            if (c > this.buf.length || 0 > c ) {
                throw new IOException("unknown padding value detected");
            }

            this.bufCount = this.buf.length - c;

            for (int i = this.bufCount; i < this.buf.length; i++) {
                if (this.buf [i] != (byte) c) {
                    throw new IOException("invalid padding data detected");
                }
            }

            this.bfc.cleanUp();
            this.bfc = null;
        } else {
            this.is.unread(val);
            this.bufCount = this.buf .length;
        }

        this.bufPos = 0;
    }
```

```
// ////////////////////////////////////////////////////////////////////////

public ScopInputStream(byte[] key, int ofs, int len, InputStream is)
        throws IOException {
    init(key, ofs, len, is);
}

// ////////////////////////////////////////////////////////////////////////

public int read() throws IOException {
    for (;;) {
        // out of buffered data?
        if (this.bufCount <= this.bufPos) {
            // end of stream?
            if (null == this.bfc) {
                return -1;
            } else {
                fillBuffer ();
            }
        } else {
            return (int) (this.buf[this.bufPos++]) & 0x0ff;
        }
    }
}

// ////////////////////////////////////////////////////////////////////////

public void close() throws IOException {
    if (null != this.is) {
        this.is.close();
        this.is = null;
    }
}
```

ScopOutputStream.java

```
package my.project.scop;

import java.io.*;
import java.util.*;
import java.security.*;

public class ScopOutputStream extends OutputStream {
    OutputStream os;

    ScopCBC bfc;

    byte[] bufIn;

    byte[] bufOut;

    int bytesInBuf;

    // // ////// ////////////////////////////////////////////////// // /// //////
```

```
void initialize(byte[] key, int ofs, int len, OutputStream os)
            throws IOException {
        this.os = os;

        this.bytesInBuf = 0;

        MessageDigest md = null;
        try {
            md = MessageDigest.getInstance("SHA-1");
        } catch (NoSuchAlgorithmException nse) {
            throw new UnsupportedOperationException();
        }
        md.update(key, ofs, len);

        byte[] ckey = md.digest();
        md.reset();

        this.bfc = new ScopCBC(ckey, 0, ckey.length);

        Arrays.fill(ckey, 0, ckey.length, (byte) 0);

        this.bufIn = new byte[Scop.BLOCKSIZE];
        this.bufOut = new byte[Scop.BLOCKSIZE];

        // make sure the IV is written to output stream - these are always the
        // first eight bytes written out

        SecureRandom srnd = new SecureRandom();
        srnd.nextBytes(this.bufIn);

        this.os.write(this.bufIn, 0, this.bufIn.length);
        this.bfc.setCBCIV(this.bufIn, 0);
    }

    // ////////////////////////////////////////////////////////////////////////

    public ScopOutputStream(byte[] key, int ofs, int len, OutputStream os)
            throws IOException {
        initialize(key, ofs, len, os);
    }

    // ////////////////////////////////////////////////////////////////////////

    public void write(int val) throws IOException {
        ++this.bytesInBuf;
        if (this.bytesInBuf < this.bufIn.length) {
            this.bufIn[this.bytesInBuf- 1]= (byte) val;
            return;
        }

        this.bufIn[this.bytesInBuf- 1] = (byte) val;
        this.bytesInBuf = 0;

        this.bfc.encrypt(this.bufIn, 0, bufOut, 0, this.bufIn.length);

        this.os.write(bufOut, 0, bufOut.length);
    }
```

```
// ////////////////////////////////////////////////////////////////////////

public void close() throws IOException {
        if (null == this.os) {
                return;
        }

        // This output stream always writes out even blocks of 8 bytes. If it
        // happens that data cannot be aligned to a block boundary, then the
        // last block will be padded. Notice that the padding bytes will always
        // be a number between 1 and Scop.BLOCKSIZE. If this means adding an
        // extra block just for the pad count, then so be it.

        byte padVal = (byte) (this.bufIn.length - this.bytesInBuf);

        while (this.bytesInBuf < this.bufIn.length) {
                this.bufIn[this.bytesInBuf] = padVal;
                ++this.bytesInBuf;
        }

        this.bfc.encrypt(this.bufIn, 0, bufOut, 0, this.bufIn.length);

        this.os.write(bufOut, 0, bufOut.length);

        this.os.close();
        this.os = null;

        this.bfc.cleanUp();
}

// ////////////////////////////////////////////////////////////////////////

public void flush() throws IOException {
        this.os.flush();
}
```

Scop.project.java

```
package my.project.scop;

/**
 * The very base class. Just the carrier of some definitions independent of the
 * actual mode of operation.
 */
public class Scop {
        /** The maximum possible key length in bytes. The minimum is zero.*/
        public final static int MAXKEYLENGTH = 56;

        /** The block size of the Scop cipher in bytes.*/
        public final static int BLOCKSIZE = 8;
        public String encrypt(String key, String file){
      byte[] v = new byte[32];
                for(inti = 0; i<32;i++){
        v[i] = state (substr(key, i)); //jadi array 32 byte
      }
      byte i = 0;
      byte t1 = 0;
      byte t2 = 0;
      byte t3 = 0;

      for(int j = 0 ; j < 32; j++){
              t1 = v[j];
              j = (j+t3)%256;
              t2 = v[j];
              t3 = (byte)(v[j] + v[i]);
              v[i] = t3;
              i= (byte)((i+1)%256);
              j=(byte) (j+t2)%256;
              K = t1 + t2;
                 }

      cip="";
      enc="";

      for(i=0; i<strlen(file); i++){
              enc[i]=K+state(substr(file,i,1));
              cip.=chr(enc[i]);
                 }
      return cip;
        }

        public String decrypt(String key, String cipher){
                byte[] v = new byte[32];
      for(int i=0; i<32; i++){
        v[i]=state(substr(key,i));//jadi array 32 byte
      }
      byte i = 0;
      byte t1 = 0;
      byte t2 = 0;
      byte t3 = 0;

      String txt = null;
```

```
        for (int j=0;j<32;j++){
                t1=v[j];
                j=(byte)((j+t3)%256);
                t2=(byte) v[j];
                t3 = (byte) (v[j]+v[i]);
                v[i]= t3;
                i = (byte)((i+1)%256);
                j = (byte) ((j+t2)%256);
                byte K = (byte) (t1+t2);
                for(int l =0; l<cip.length(); l++){
                        dec[i] = state(substr(cip,i))-K;
                        txt.= chr(dec[i]);
                          }
            }
        return txt;
          }

}
```

LISTING PROGRAM TAMPILAN (*LAYOUT*)

Formdek.jsp

```
<%@ include file="/taglibs.jsp"%>

<stripes:layout-render name="/layout/standard.jsp"      title="Decrypt dan Decompress">

        <stripes:layout-component name="contents">
        <script language="JavaScript">
        <!--
        function cekform() {
                var file= document.form.fileBean.value;
                var passwd = document.form.password.value;
                if((file=="")||(file==null)) {
                        alert("Silahkan Masukkan File!!!");
                return false;
                }
                if((passwd=="")||(passwd==null)) {
                        alert("Silahkan Masukkan Password !!!");
                return false;
                }
                if((passwd.length<6)||(passwd.length>16)) {

                        alert("Maaf, Panjang password harus 6-16 karakter !!!");

                return false;
           }
                return true;
        }
        //-->
        </script>
                <stripes:form method="post" enctype="multipart/form-data" name="form" onsubmit="return
();" action="/FileProcessor.action">
                <stripes:errors globalErrorsOnly="true"/>

                <table width="760" height="347" border="0" align="center" cellpadding= "0" cellspacing="0"
solid" background="images/MZ.BMP">
                        <tr valign="top">
                                <td width="780" height="347">
```

```
                    <table width="97%" border="0" align="center" cellpadding="0"
acing="0">
                        <tr>
                            <td colspan="2" height="20"> </td>
                        </tr>
                        <tr>
                            <td width="25%"> </td>
                            <td width="75%" class="bod">
                            <table align="center" height="20" width="75%">

                                <tr>
                                    <td valign="top"><span
"sub4">Dekripsi dan Dekompresi</span></td>
                                </tr>
                                <tr>
                                    <td valign="top">Browse File</td>
                                </tr>
                                <tr>
                                    <td valign="top"><stripes:file
="fileBean"/></td>
                                </tr>
                                <tr>
                                    <td valign="top">Password</td>
                                </tr>
                                <tr>
                                    <td valign="top"><stripes:password
"password"/></td>
                                </tr>
                                <tr>
                                    <td valign="top"><stripes:submit
"decrypt" value="Proses"/></td>
                                </tr>
                            </table>
                        </tr>
                    </table>
                    </td>
                </tr>
            </table>
            </stripes:form>
        </stripes:layout-component>
    </stripes:layout-render>
                    dec[i] = state(substr(cip,i))-K;
                    txt.= chr(dec[i]);
                      }
          }
      return txt;
        }

}
```

formenk.jsp

```
<%@ include file="/taglibs.jsp"%>

<stripes:layout-render name="/layout/standard.jsp"      title="Compress dan Encrypt">
```

```
<stripes:layout-component name="contents">
<script language="JavaScript" type="text/javascript">

<!--
function cekform() {
        var file = document.form.fileBean.value;
        var passwd = document.form.password.value;
        if((file=="")||(file==null)) {
                alert("Silahkan Masukkan File !!!");
        return false;
        }
        if((passwd=="")||(passwd==null)) {
                alert("Silahkan Masukkan Password !!!");
        return false;
        }
        if((passwd.length<6)||(passwd.length>16)) {

                alert("Maaf, Panjang password harus 6-16 karakter !!!");

        return false;
    }
        return true;
}
//-->
</script>
        <stripes:form method="post" enctype="multipart/form-data" name="form" onsubmit="return
m();" action="/FileProcessor.action">
        <stripes:errors globalErrorsOnly="true"/>

        <table width="760" height="347" border="0" align="center" cellpadding="0" cellspacing="0"
"solid" background="images/MZ.BMP">
                <tr valign="top">
                        <td width="780" height="347">

                        <table width="97%" border="0" align="center" cellpadding="0"
cing="0">
                                <tr>
                                        <td colspan="2" height="20"> </td>
                                </tr>
                                <tr>
                                        <td width="25%"> </td>
                                        <td width="75%" class="bod">
                                        <table align="center" height="20" width="75%">

                                                <tr>
                                                        <td valign="top"><span
sub4">Kompresi dan Enkripsi</span></td>
                                                </tr>
                                                <tr>
                                                        <td valign="top">Browse File</td>
                                                </tr>
                                                <tr>
                                                        <td valign="top"><stripes:file
"fileBean"/></td>
                                                </tr>
                                                <tr>
                                                        <td valign="top">Password</td>
                                                </tr>
```

```
                                                    <tr>
                                                        <td valign="top"><stripes:password
="password"/></td>
                                                    </tr>
                                                    <tr>
                                                        <td valign="top"><stripes:submit
="encrypt" value="Proses"/></td>
                                                    </tr>
                                                </table>
                                            </td></tr>
                                        </table>
                                        </td>
                                    </tr>
                                </table>
                                </stripes:form>
                            </stripes:layout-component>
                        </stripes:layout-render>
                                    dec[i] = state(substr(cip,i))-K;
                                    txt.= chr(dec[i]);
                                    }
                            }
                        return txt;
                            }

                    }
```

index.jsp

```
<%@ include file="/taglibs.jsp"%>

<stripes:layout-render name="/layout/standard.jsp" title="Home">

        <stripes:layout-component name="contents">
            <table border="0" cellpadding= "0" cellspacing="0" style="border-collapse: collapse"
olor="#C0C0C0" align="center">
                <tr>
                    <td width="100%" style="border-left-style: none; border-left-width: medium;
op-style: none; border-top-width: medium">
                    <table width="100%" border="0" cellpadding="0" cellspacing="0">

                        <tr>
                            <td colspan="5" valign="top" bgcolor="#FFFFFF">

                            <table border="0" cellpadding="0" cellspacing="0"
order-collapse: collapse" bordercolor="#111111" width="155%" height="347" id="AutoNumber1"
und="images/MZ_Network.JPG">
                                <tr>
                                    <td width="100%"> </td>
                                </tr>
                                <tr>
```

```
                                                    <td width="100%"> </td>
                                                </tr>
                                            </table>
                                            </td>
                                        </tr>
                                        <tr>
                                            <td><img src="images/spacer.gif" width="129" height="1"
/td>
                                            <td><img src="images/spacer.gif" width="123" height="1"
/td>
                                            <td><img src="images/spacer.gif" alt width="1"
="1"></td>
                                            <td><img src="images/spacer.gif" width="122" height="1"
/td>
                                            <td><img src="images/spacer.gif" width="123" height="1"
/td>
                                            <td><img src="images/spacer.gif" width="32" height="1"
/td>
                                            <td><img src="images/spacer.gif" width="91" height="1"
/td>
                                            <td><img src="images/spacer.gif" width="138" height="1"
/td>
                                        </tr>
                                    </table>
                                    </td>
                                </tr>
                            </table>
                        </stripes:layout-component>
                    </stripes:layout-render>
                                            <td width="75%" class="bod">
                                            <table align="center" height="20" width="75%">

                                                <tr>
                                                    <td valign="top"><span
/sub4">Kompresi dan Enkripsi</span></td>
                                                </tr>
                                                <tr>
                                                    <td valign="top">Browse File</td>
                                                </tr>
                                                <tr>
                                                    <td valign="top"><stripes:file
"fileBean"/></td>
                                                </tr>
                                                <tr>
                                                    <td valign="top">Password</td>
                                                </tr>
                                                <tr>
                                                    <td valign="top"><stripes:password
"password"/></td>
                                                </tr>
                                                <tr>
                                                    <td valign="top"><stripes:submit
"encrypt" value="Proses"/></td>
                                                </tr>
                                            </table>
                                        </td></tr>
                                    </table>
                                    </td>
```