```
    if(mode != 1)
      {
        rc=serpent_convert_from_string(cipher.blockSize, IV, cipher.IVi);
        for(int i=0; i<4; i++){
              cipher.IVb[i*4+0] = (byte)cipher.IVi[i];
              cipher.IVb[i*4+1] = (byte)(cipher.IVi[i]>>>8);
              cipher.IVb[i*4+2] = (byte)(cipher.IVi[i]>>>16);
              cipher.IVb[i*4+3] = (byte)(cipher.IVi[i]>>>24);
        }
        if(rc<=0)
          return -5;
      }

    return 1;
}




int blockEncrypt(cipherInstance cipher,
                         keyInstance key,
                         int[] input,
                         int inputLen,
                         int[] outBuffer)
{
    int[] t = new int[4];
    int[] u = new int[4];
    int    b,    n, i;
    int bit, bit0, ctBit, carry;

    /*
     * Note about optimization: the code becomes slower of the calls to
     * serpent_encrypt and serpent_decrypt are replaced by inlined code.
     * (tested on Pentium 133MMX)
     */

    switch(cipher.mode)
      {
      case 1:

          for(b=0 ; b<inputLen; b+=128 ){
                    if(obmode != 3){
                              u[0] = pbuf[oip+0];
                              u[1] = pbuf[oip+1];
                              u[2] = pbuf[oip+2];
                              u[3] = pbuf[oip+3];
                    }
                    if(obmode == 3){
                              u[0] = cipher.IVi[0];
                    u[1] = cipher.IVi[1];
                    u[2] = cipher.IVi[2];
```

```
                u[3] = cipher.IVi[3];
                }
                serpent_encrypt( u, 0, x, 0, key.subkeys);
                if(obmode != 3){
                        cbuf[oip+0]=x[0];
                        cbuf[oip+1]=x[1];
                        cbuf[oip+2]=x[2];
                        cbuf[oip+3]=x[3];
                        for(i=0; i<4; i++){
                                cbufb[4*(oip+i)+0] = (byte)(cbuf[oip+i]);
                                cbufb[4*(oip+i)+1] = (byte)(cbuf[oip+i]>>>8);
                                cbufb[4*(oip+i)+2] = (byte)(cbuf[oip+i]>>>16);
                                cbufb[4*(oip+i)+3] = (byte)(cbuf[oip+i]>>>24);
                        }
                }
                oip+=4;
        }
        return inputLen;
case 2:
        t[0] = cipher.IVi[0];
        t[1] = cipher.IVi[1];
        t[2] = cipher.IVi[2];
        t[3] = cipher.IVi[3];
        for(b=0; b<inputLen; b+=128)
        {
                t[0] ^= pbuf[oip+0];
                t[1] ^= pbuf[oip+1];
                t[2] ^= pbuf[oip+2];
                t[3] ^= pbuf[oip+3];
                serpent_encrypt(t,0, t,0, key.subkeys);
                cbuf[oip+0] = t[0];
                cbuf[oip+1] = t[1];
                cbuf[oip+2] = t[2];
                cbuf[oip+3] = t[3];
                for(i=0; i<4; i++){
                        cbufb[4*(oip+i)+0] = (byte)(cbuf[oip+i]);
                        cbufb[4*(oip+i)+1] = (byte)(cbuf[oip+i]>>>8);
                        cbufb[4*(oip+i)+2] = (byte)(cbuf[oip+i]>>>16);
                        cbufb[4*(oip+i)+3] = (byte)(cbuf[oip+i]>>>24);
                }
                oip += 4;
        }
        cipher.IVi[0] = t[0];
        cipher.IVi[1] = t[1];
        cipher.IVi[2] = t[2];
        cipher.IVi[3] = t[3];

        return inputLen;

case 3:
                cipher.mode = 1;  /* do encryption in ECB */
```

```
                        for (n=0;n<inputLen;n++)
                        {
                                blockEncrypt(cipher,key,cipher.IVi,128,x);
                                bit0    =   (0x80 >>> (n & 7));/* which bit position in byte */
                                ctBit =   ((cbufb[n/8] & bit0) ^ (((byte)(x[0]) & 0x80) >>> (n&7)));
                                pbufb[n/8] =    (byte) ((pbufb[n/8] & ~ bit0) | ctBit);
//                              carry =    (ctBit >>> (7 - (n&7)));
                                int ti = ctBit;
                                if(ti<0){
                                        ctBit ^= 0x80;
                                        ti = ctBit;
                                        ti += 0x80;
                                        }
                                carry = (byte) (ti >>> (7 - (n&7)));
                                for (i=128/8-1;i>=0;i--)
                                        {
                                        bit =   (cipher.IVb[i] >>> 7);          /* save next "carry" from
shift */
                                        if(bit < 0){bit = 1;}
                                        cipher.IVb[i] = (byte) ((cipher.IVb[i] << 1) ^ carry);
                                        carry = bit;
                                        }
                                int jj=0;
                                for(i=0; i*4<128/8; i++){
                                        for (int p = 0; p < 4; p++) {
                                                int tmpi = (int)(cipher.IVb[i*4+3-p] & 0xff);
                                                if(tmpi < 0){
                                                        cipher.IVb[i*4+3-p] ^= 0x80;
                                                        tmpi = cipher.IVb[i*4+3-p];
                                                        tmpi += 0x80;
                                                }
                                                jj = (jj << 8) | tmpi;
                                        }
                                        cipher.IVi[i] = jj;
                                        jj = 0;
                                }
                        }
                        cipher.mode = 3; /* restore mode for next time */
                        return inputLen;

                default:
                   return -5;
                }
        }



        int blockDecrypt(cipherInstance cipher,
                         keyInstance key,
                         int[] input,
                         int inputLen,
```

```
                        int[] outBuffer)
{
   int[] t = new int[4];
   int[] u = new int[4];
   int[] v = new int[4];
   int    b, n, i;
   int bit, bit0, ctBit, carry;

   switch(cipher.mode)
      {
      case 1:
         for(b=0; b<inputLen; b+=128){
               if(obmode != 3){
                        u[0]=cbuf[oip+0];
                        u[1]=cbuf[oip+1];
                        u[2]=cbuf[oip+2];
                        u[3]=cbuf[oip+3];
               }
               if(obmode==3){
                        u[0] = cipher.IVi[0];
                     u[1] = cipher.IVi[1];
                     u[2] = cipher.IVi[2];
                     u[3] = cipher.IVi[3];
               }
               serpent_decrypt( u, 0, x, 0, key.subkeys);
               if(obmode != 3){
                        pbuf[oip+0]=x[0];
                        pbuf[oip+1]=x[1];
                        pbuf[oip+2]=x[2];
                        pbuf[oip+3]=x[3];
                        for(i=0; i<4; i++){
                                 pbufb[4*(oip+i)+0] = (byte)(pbuf[oip+i]);
                                 pbufb[4*(oip+i)+1] = (byte)(pbuf[oip+i]>>>8);
                                 pbufb[4*(oip+i)+2] = (byte)(pbuf[oip+i]>>>16);
                                 pbufb[4*(oip+i)+3] = (byte)(pbuf[oip+i]>>>24);
                        }
               }
               oip += 4;
         }
         return inputLen;

      case 2:
         t[0] = cipher.IVi[0];
         t[1] = cipher.IVi[1];
         t[2] = cipher.IVi[2];
         t[3] = cipher.IVi[3];
         for(b=0; b<inputLen; b+=128)
         {
               u[0]=cbuf[oip+0];
               u[1]=cbuf[oip+1];
               u[2]=cbuf[oip+2];
```

```
                u[3]=cbuf[oip+3];
                serpent_decrypt(u,0, v, 0, key.subkeys);
                v[0] ^= t[0];
                v[1] ^= t[1];
                v[2] ^= t[2];
                v[3] ^= t[3];
                t[0] = u[0];
                t[1] = u[1];
                t[2] = u[2];
                t[3] = u[3];
                pbuf[oip+0]=v[0];
                pbuf[oip+1]=v[1];
                pbuf[oip+2]=v[2];
                pbuf[oip+3]=v[3];
                for(i=0; i<4; i++){
                        pbufb[4*(oip+i)+0] = (byte)(pbuf[oip+i]);
                        pbufb[4*(oip+i)+1] = (byte)(pbuf[oip+i]>>>8);
                        pbufb[4*(oip+i)+2] = (byte)(pbuf[oip+i]>>>16);
                                pbufb[4*(oip+i)+3] = (byte)(pbuf[oip+i]>>>24);
                }
                oip += 4;
        }
        cipher.IVi[0] = t[0];
        cipher.IVi[1] = t[1];
        cipher.IVi[2] = t[2];
        cipher.IVi[3] = t[3];

        return inputLen;

    case 3://blockDecrypt
                cipher.mode = 1; /* do encryption in ECB */
                obmode = 3;
                for (n=0;n<inputLen;n++)
                {
                        blockEncrypt(cipher, key, cipher.IVi, 128, x);
                        for(i=0; i<4; i++){
                                cipher.IVb[i*4+0] = (byte)cipher.IVi[i];
                                cipher.IVb[i*4+1] = (byte)(cipher.IVi[i]>>>8);
                                cipher.IVb[i*4+2] = (byte)(cipher.IVi[i]>>>16);
                                cipher.IVb[i*4+3] = (byte)(cipher.IVi[i]>>>24);
                        }
                        bit0    =   (0x80 >>> (n & 7));
                        ctBit =   (cbufb[n/8] & bit0);
                        pbufb[n/8] =    (byte) ((pbufb[n/8] & ~ bit0) |
                                                        (ctBit  ^  (((byte)(x[0])  &  0x80)
>>> (n&7))));
//                      carry =    (ctBit >>> (7 - (n&7)));
                        int ti = ctBit;
                        if(ti<0){
                                ctBit ^= 0x80;
                                ti = ctBit;
```

```
                                        ti += 0x80;
                                        }
                                carry = (byte) (ti >>> (7 - (n&7)));
                                for (i=128/8-1;i>=0;i--)
                                        {
                                        bit =   (cipher.IVb[i] >>> 7);          /* save next "carry" from
shift */
                                        if(bit < 0){bit = 1;}
                                        cipher.IVb[i] =   (byte) ((cipher.IVb[i] << 1) ^ carry);
                                        carry = bit;
                                        }
                                int jj=0;
                                for(i=0; i*4<128/8; i++){
                                        for (int p = 0; p < 4; p++) {
                                                int tmpi = (int)(cipher.IVb[i*4+3-p] & 0xff);
                                                if(tmpi < 0){
                                                        cipher.IVb[i*4+3-p] ^= 0x80;
                                                        tmpi = cipher.IVb[i*4+3-p];
                                                        tmpi += 0x80;
                                                }
                                                jj = (jj << 8) | tmpi;
                                        }
                                        cipher.IVi[i] = jj;
                                        jj = 0;
                                }
                        }
                        cipher.mode = 3; /* restore mode for next time */
                        obmode = 0;
                        return inputLen;

                default:
                   return -5;
                }
        }

        void serpent_encrypt(int[] plaintext, int ps,
                        int[] ciphertext, int cs,
                        int[][] subkeys)
        {
           int x0=0, x1=1, x2=2, x3=3;
           int y0=0, y1=1, y2=2, y3=3;
           int[] y = new int[4];
           int[] x = new int[4];

           x[x0]=plaintext[ps+0];
           x[x1]=plaintext[ps+1];
           x[x2]=plaintext[ps+2];
           x[x3]=plaintext[ps+3];

           /* Start to encrypt the plaintext x */
           keying(x, x0, x1, x2, x3, subkeys[ 0]);
```

```
RND00(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
keying(x, x0, x1, x2, x3, subkeys[ 1]);
RND01(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
keying(x, x0, x1, x2, x3, subkeys[ 2]);
RND02(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
keying(x, x0, x1, x2, x3, subkeys[ 3]);
RND03(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
keying(x, x0, x1, x2, x3, subkeys[ 4]);
RND04(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
keying(x, x0, x1, x2, x3, subkeys[ 5]);
RND05(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
keying(x, x0, x1, x2, x3, subkeys[ 6]);
RND06(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
keying(x, x0, x1, x2, x3, subkeys[ 7]);
RND07(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
keying(x, x0, x1, x2, x3, subkeys[ 8]);
RND08(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
keying(x, x0, x1, x2, x3, subkeys[ 9]);
RND09(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
keying(x, x0, x1, x2, x3, subkeys[10]);
RND10(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
keying(x, x0, x1, x2, x3, subkeys[11]);
RND11(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
keying(x, x0, x1, x2, x3, subkeys[12]);
RND12(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
keying(x, x0, x1, x2, x3, subkeys[13]);
RND13(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
keying(x, x0, x1, x2, x3, subkeys[14]);
RND14(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
keying(x, x0, x1, x2, x3, subkeys[15]);
RND15(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
keying(x, x0, x1, x2, x3, subkeys[16]);
RND16(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
keying(x, x0, x1, x2, x3, subkeys[17]);
```

```
RND17(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
keying(x, x0, x1, x2, x3, subkeys[18]);
RND18(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
keying(x, x0, x1, x2, x3, subkeys[19]);
RND19(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
keying(x, x0, x1, x2, x3, subkeys[20]);
RND20(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
keying(x, x0, x1, x2, x3, subkeys[21]);
RND21(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
keying(x, x0, x1, x2, x3, subkeys[22]);
RND22(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
keying(x, x0, x1, x2, x3, subkeys[23]);
RND23(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
keying(x, x0, x1, x2, x3, subkeys[24]);
RND24(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
keying(x, x0, x1, x2, x3, subkeys[25]);
RND25(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
keying(x, x0, x1, x2, x3, subkeys[26]);
RND26(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
keying(x, x0, x1, x2, x3, subkeys[27]);
RND27(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
keying(x, x0, x1, x2, x3, subkeys[28]);
RND28(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
keying(x, x0, x1, x2, x3, subkeys[29]);
RND29(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
keying(x, x0, x1, x2, x3, subkeys[30]);
RND30(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
keying(x, x0, x1, x2, x3, subkeys[31]);
RND31(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
x[x0] = y[y0]; x[x1] = y[y1]; x[x2] = y[y2]; x[x3] = y[y3];
keying(x, x0, x1, x2, x3, subkeys[32]);
/* The ciphertext is now in x */

ciphertext[cs+0] = x[x0];
ciphertext[cs+1] = x[x1];
ciphertext[cs+2] = x[x2];
ciphertext[cs+3] = x[x3];
```

```
}

void serpent_decrypt(int[] ciphertext, int cs,
                     int[] plaintext, int ps,
                     int[][] subkeys)
{
   int x0=0, x1=1, x2=2, x3=3;
   int y0=0, y1=1, y2=2, y3=3;
   int[] x = new int[4];
   int[] y = new int[4];

   x[x0]=ciphertext[cs+0];
   x[x1]=ciphertext[cs+1];
   x[x2]=ciphertext[cs+2];
   x[x3]=ciphertext[cs+3];

   /* Start to decrypt the ciphertext x */
   keying(x, x0, x1, x2, x3, subkeys[32]);
   InvRND31(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
   keying(y, y0, y1, y2, y3, subkeys[31]);
   inv_transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
   InvRND30(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
   keying(y, y0, y1, y2, y3, subkeys[30]);
   inv_transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
   InvRND29(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
   keying(y, y0, y1, y2, y3, subkeys[29]);
   inv_transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
   InvRND28(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
   keying(y, y0, y1, y2, y3, subkeys[28]);
   inv_transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
   InvRND27(x, x0, x1, x2, x3,y,    y0, y1, y2, y3);
   keying(y, y0, y1, y2, y3, subkeys[27]);
   inv_transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
   InvRND26(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
   keying(y, y0, y1, y2, y3, subkeys[26]);
   inv_transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
   InvRND25(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
   keying(y, y0, y1, y2, y3, subkeys[25]);
   inv_transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
   InvRND24(x, x0, x1, x2, x3,y,    y0, y1, y2, y3);
   keying(y, y0, y1, y2, y3, subkeys[24]);
   inv_transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
   InvRND23(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
   keying(y, y0, y1, y2, y3, subkeys[23]);
   inv_transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
   InvRND22(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
   keying(y, y0, y1, y2, y3, subkeys[22]);
   inv_transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
   InvRND21(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
   keying(y, y0, y1, y2, y3, subkeys[21]);
   inv_transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
```

```
InvRND20(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
keying(y, y0, y1, y2, y3, subkeys[20]);
inv_transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
InvRND19(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
keying(y, y0, y1, y2, y3, subkeys[19]);
inv_transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
InvRND18(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
keying(y, y0, y1, y2, y3, subkeys[18]);
inv_transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
InvRND17(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
keying(y, y0, y1, y2, y3, subkeys[17]);
inv_transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
InvRND16(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
keying(y, y0, y1, y2, y3, subkeys[16]);
inv_transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
InvRND15(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
keying(y, y0, y1, y2, y3, subkeys[15]);
inv_transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
InvRND14(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
keying(y, y0, y1, y2, y3, subkeys[14]);
inv_transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
InvRND13(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
keying(y, y0, y1, y2, y3, subkeys[13]);
inv_transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
InvRND12(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
keying(y, y0, y1, y2, y3, subkeys[12]);
inv_transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
InvRND11(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
keying(y, y0, y1, y2, y3, subkeys[11]);
inv_transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
InvRND10(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
keying(y, y0, y1, y2, y3, subkeys[10]);
inv_transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
InvRND09(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
keying(y, y0, y1, y2, y3, subkeys[ 9]);
inv_transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
InvRND08(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
keying(y, y0, y1, y2, y3, subkeys[ 8]);
inv_transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
InvRND07(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
keying(y, y0, y1, y2, y3, subkeys[ 7]);
inv_transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
InvRND06(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
keying(y, y0, y1, y2, y3, subkeys[ 6]);
inv_transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
InvRND05(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
keying(y, y0, y1, y2, y3, subkeys[ 5]);
inv_transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
InvRND04(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
keying(y, y0, y1, y2, y3, subkeys[ 4]);
inv_transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
```

```
        InvRND03(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
        keying(y, y0, y1, y2, y3, subkeys[ 3]);
        inv_transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
        InvRND02(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
        keying(y, y0, y1, y2, y3, subkeys[ 2]);
        inv_transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
        InvRND01(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
        keying(y, y0, y1, y2, y3, subkeys[ 1]);
        inv_transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
        InvRND00(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
        x[x0] = y[y0]; x[x1] = y[y1]; x[x2] = y[y2]; x[x3] = y[y3];
        keying(x, x0, x1, x2, x3, subkeys[ 0]);
        /* The plaintext is now in x */

        plaintext[ps+0] = x[x0];
        plaintext[ps+1] = x[x1];
        plaintext[ps+2] = x[x2];
        plaintext[ps+3] = x[x3];
    }

//  #define min(x,y) (((x)<(y))?(x):(y))

    int serpent_convert_from_string(int len, int[] str, int[] val)
    /* the size of val must be at least the next multiple of 32 */
    /* bits after len bits */
    {
        int is, iv, i, j, k;
        byte[] tmpch4 = new byte[4];
        int tmpi =0, jj = 0;
        int slen = (((str.length)<((len+3)/4))?(str.length):((len+3)/4));// min(str.length, (len+3)/4);

        if(len<0)
            return -1;              /* Error!!! */

        if(len>slen*4 || len<slen*4-3)
            return -1;              /* Error!!! */

        for(is=0; is<slen; is++)
            if(((str[is]<'0')||(str[is]>'9')) &&
               ((str[is]<'A')||(str[is]>'F')) &&
               ((str[is]<'a')||(str[is]>'f')))
                return -1;  /* Error!!! */

        for(is=slen, iv=0; is>=8; is-=8, iv++)
            {
                byte t;
//              sscanf(&str[is-8], "%08lX", &t);
                    for( i=0; i<4; i++){
                            j =   str[is-2*i-2];
                            k =   str[is-2*i-1];
                            if(j>=0x30 && j<=0x39) j =   (j-0x30);
```

```
                    else{
                            if(j>=0x41 && j<=0x46) j =   (j-0x41+0x0A);
                            if(j>=0x61 && j<=0x66) j =   (j-0x61+0x0A);
                    }
                    if(k>=0x30 && k<=0x39) k =   (k-0x30);
                    else{
                            if(k>=0x41 && k<=0x46) k =   (k-0x41+0x0A);
                            if(k>=0x61 && k<=0x66) k =   (k-0x61+0x0A);
                    }
                    tmpch4[i] =   (byte) (j*0x10 + k);
            }
            for (int p = 0; p < tmpch4.length; p++) {
                tmpi = (int)(tmpch4[3-p] & 0xff);
                if(tmpi < 0){
                        tmpch4[3-p] ^= 0x80;
                        tmpi = tmpch4[3-p];
                        tmpi += 0x80;
                }
                jj = (jj << 8) | tmpi;
            }
                    val[iv] = jj;
      }
    if(is>0)
    {
            byte[] tmp = new byte[10];
        byte t;
//          strncpy(tmp, str, is);
        for(i=0; i<is; i++){
            tmp[i] = (byte) str[i];
        }
        tmp[is] = 0;
//          sscanf(tmp, "%08lX", &t);
            for( i=0; i<4; i++){
                    j =   str[is-8+2*i];
                    k =   str[is-8+2*i+1];
                    if(j>=0x30 && j<=0x39) j =   (j-0x30);
                    else{
                            if(j>=0x41 && j<=0x46) j =   (j-0x41+0x0A);
                            if(j>=0x61 && j<=0x66) j =   (j-0x61+0x0A);
                    }
                    if(k>=0x30 && k<=0x39) k =   (k-0x30);
                    else{
                            if(k>=0x41 && k<=0x46) k =   (k-0x41+0x0A);
                            if(k>=0x61 && k<=0x66) k =   (k-0x61+0x0A);
                    }
                    tmpch4[i] =   (byte) (j*0x10 + k);
            }

            tmpi =0; jj = 0;
            for (int p = 0; p < tmpch4.length; p++) {
                    tmpi = (int)(tmpch4[p] & 0xff);
```

```
                    if(tmpi < 0){
                        tmpch4[p] ^= 0x80;
                        tmpi = tmpch4[p];
                        tmpi += 0x80;
                    }
                    jj = (jj << 8) | tmpi;
                }
                val[iv++] = jj;
    }
    for(; iv<(len+31)/32; iv++)
        val[iv] = 0;
    return iv;
}

byte toChar( byte c )
{
        if( c >= 0 && c <= 9 )                          // 0～9 ならば
                return (byte) ( c + 0x30 ); // ASCII に変換して返す
        else if( c >= 10 && c <= 15 )           // 10～15 ならば
                return (byte) ( c + 0x37 ); // A～F の ASCII を返す
        else
                return ' ';
}

byte[] serpent_convert_to_string(int len, int[] val, byte[] str)
/* str must have at least (len+3)/4+1 bytes. */
{
        int i, j, k=0;
        byte[] tmp = new byte[10];
        if(len<0){
                str[0] = '0';
                return str;                     /* Error!!! */
        }

        str[0] = 0;
        i=len/32;
        if((len&31)>0)
        {
                //                      byte[] tmp = new byte[10];
                //                      sprintf(tmp, "%08lX", val[i]&(((len&31)<<1)-1));

                        j = val[i]&(((len&31)<<1)-1);
                        tmp[0] = (byte)(j);
                        tmp[2] = (byte)(j>>>8);
                        tmp[4] = (byte)(j>>>16);
                        tmp[6] = (byte)(j>>>24);

                        tmp[0] = (byte) (tmp[0]&0x0f);
                        tmp[1] = (byte) (tmp[0]>>>4);
                        tmp[2] = (byte)(tmp[2]&0x0f);
                        tmp[3] = (byte)(tmp[2]>>>4);
```

```
		tmp[4] = (byte) (tmp[4]&0x0f);
		tmp[5] = (byte) (tmp[4]>>>4);
		tmp[6] = (byte)(tmp[6]&0x0f);
		tmp[7] = (byte)(tmp[6]>>>4);

		for(i=0; i<8 ; i++){
			tmp[i] = toChar(tmp[i]);
		}

		//			strcat(str, &tmp[8-(((len&31)+3)/4)]);

		k = 0;
		for(i=0; i<str.length; i++){
			if(str[i] == 0){k = i;}
		}
		for(i=0; i<(((len&31)+3)/4); i++){
			str[k+i] = tmp[i+8-(((len&31)+3)/4)];
		}
	}
	for(i--; i>=0; i--)
	{

		//			sprintf(tmp, "%08lX", val[i]);
		//			strcat(str, tmp);

		j = val[i];
		tmp[0] = (byte)(j);
		tmp[2] = (byte)(j>>>8);
		tmp[4] = (byte)(j>>>16);
		tmp[6] = (byte)(j>>>24);

		tmp[0] = (byte) (tmp[0]&0x0f);
		tmp[1] = (byte) (tmp[0]>>>4);
		tmp[2] = (byte)(tmp[2]&0x0f);
		tmp[3] = (byte)(tmp[2]>>>4);
		tmp[4] = (byte) (tmp[4]&0x0f);
		tmp[5] = (byte) (tmp[4]>>>4);
		tmp[6] = (byte)(tmp[6]&0x0f);
		tmp[7] = (byte)(tmp[6]>>>4);

		for(i=0; i<8 ; i++){
			tmp[i] = toChar(tmp[i]);
		}

		k = 0;
		for(i=0; i<8; i++){
			if(str[i] == 0){
				k = i;
				break;
			}
		}
```

```
                }
                for(i=0; i<(((len&31)+3)/4); i++){
                        str[k+i] = tmp[i];
                }
                return str;
        }



        int s;
//      keyInstance keyI;
//      cipherInstance cipherI;

////////////////////////////////////////////////////////
        void SerpentDC(String keyfn, String ctfn, String ptfn)                  // 引数へのポインタ
        {
                int i;
                byte[] c_mode = new byte[3];
                byte[] c_klen = new byte[5];
                int[] c_key = new int[64+2];
                byte[] c_keyb = null;//new byte[64+2];
                int[] c_cini = new int[32+2];
                byte[] c_cinib = new byte[32+2];
            int len, rlen, blen4, pfilelen;
                int mode,klen,blen,rc=0;

                blen4 = 2048;

                cipherInstance cipherI = new cipherInstance();
                keyInstance keyI = new keyInstance();
                ////////////////////////////////////////////////

                try{
//                      FileOutputStream foutst = openFileOutput(ptfn,MODE_PRIVATE);
//                      FileInputStream finst = openFileInput(ctfn);

                        File fin = new File(ctfn);
                        fin.getParentFile().mkdir();
                        FileInputStream finst=null;
                        try {
                                finst = new FileInputStream(fin);
                        } catch (FileNotFoundException e5) {
                                // TODO 自動生成された catch ブロック
                                e5.printStackTrace();
                        }

                        File fout = new File(ptfn);
                        fout.getParentFile().mkdir();
                        FileOutputStream foutst=null;
                        try {
                                foutst = new FileOutputStream(fout);
```

```
                } catch (FileNotFoundException e5) {
                        // TODO 自動生成された catch ブロック
                        e5.printStackTrace();
                }


                File fkey = new File(keyfn);
                fkey.getParentFile().mkdir();
                FileInputStream inkeyst=null;
                try {
                        inkeyst = new FileInputStream(fkey);
                        inkeyst.read(c_mode);
                        inkeyst.read(c_klen);
                        klen = atoi(c_klen);
                        c_keyb = new byte[klen/4+2];
                        inkeyst.read(c_keyb);
                        inkeyst.read(c_cinib);
                } catch (IOException e3) {
                        // TODO 自動生成された catch ブロック
                        e3.printStackTrace();
                }//127

                mode = atoi(c_mode);
                klen = atoi(c_klen);
                blen = 128;

                for(i=0; i<32 ; i++){
                        c_cini[i] = c_cinib[i];
                }

                if(klen<56 || 256<klen){
//                      printf("Wrong key size. ¥n");
                        return ;
                }

                /*Set mode*/
                if(mode == 1){
                        int[] tmpb = new int[1];
                        tmpb[0] = ' ';
                        rc=cipherInit(cipherI, 1, tmpb);
                }
                if(mode == 2){
                        rc=cipherInit(cipherI, 2, c_cini);
                }
                if(mode == 3){
                        rc=cipherInit(cipherI, 3, c_cini);
                }
                if(rc<=0){
//                      printf("モード設定が出来ません。");
                        return;
                }
```

```
        for(i=0; i<klen/4;i++){
                c_key[i] = c_keyb[i];
        }
        serpent_makeKey(keyI, 1,    klen, c_key );

int flen = finst.available();
rlen = flen;

        s = 4;//sizeof(unsigned long);
        // write the bytes of the file
        if(blen4<=flen){
                len = finst.read(cbufb, 0, blen4 );
        }else{
                len = finst.read(cbufb, 0, flen );
        }
        rlen = rlen - len;

        if(len < blen4){ return ;    }

        int jj =0;
        for(i=0; i*4<len; i++){
                for(int p=0; p<4; p++){
                        int tmp = cbufb[i*4+3-p];
                        if(tmp < 0){
                                byte tmpb = (byte) (cbufb[i*4+3-p] ^ 0x80);
                                tmp = tmpb;
                                tmp += 0x80;
                        }
                        jj = (jj << 8) | tmp;
                }
                cbuf[i] = jj;
                jj = 0;
        }

        ob=0; oip=0;
        rc=blockDecrypt(cipherI, keyI, cbuf, 8*flen, pbuf);
                // 復号文出力
//      pfilelen = *((long*)(pbuf));
        byte[] tmpch4 = new byte[4];
        tmpch4[0] = pbufb[0];//(byte) pbuf[0];
        tmpch4[1] = pbufb[1];//(byte) (pbuf[0]>>>8);
        tmpch4[2] = pbufb[2];//(byte) (pbuf[0]>>>16);
        tmpch4[3] = pbufb[3];//(byte) (pbuf[0]>>>24);
        jj = 0;
        int tmp = 0;
        for (int p = 0; p < tmpch4.length; p++) {
                tmp = (tmpch4[3-p] & 0xff);
                if(tmp < 0){
                        tmpch4[3-p] ^= 0x80;
                        tmp = tmpch4[3-p];
```

```
                    tmp += 0x80;
                }
                jj = (jj << 8) | tmp;
        }
        pfilelen = jj;

        if(pfilelen <= blen4 - s){
                foutst.write(pbufb, s, pfilelen);
                if(inkeyst != null)
                {
                        try {
                                inkeyst.close();
                        } catch (IOException e) {
                                // TODO 自動生成された catch ブロック
                                e.printStackTrace();
                        }
                }

                if(foutst != null)
                {
                        try {
                                foutst.close();
                        } catch (IOException e) {
                                // TODO 自動生成された catch ブロック
                                e.printStackTrace();
                        }
                }

                if(finst != null)
                {
                        try {
                                finst.close();
                        } catch (IOException e) {
                                // TODO 自動生成された catch ブロック
                                e.printStackTrace();
                        }
                }
                return;// 0;
        }
        else{
                foutst.write(pbufb, s, blen4 - s);
                pfilelen -= (blen4 - s);
        }

if((rlen <= blen4) && (rlen > 0))
        {       // if the file length is less than or equal to 2048 bytes
        len = finst.read(cbufb, 1, blen4 );
        rlen -= len;
        if(rlen > 0){ return; }
                rc=blockDecrypt(cipherI, keyI, cbuf, 8*blen4, pbuf);
                foutst.write(pbufb, 1, pfilelen) ;
```

```
                if(inkeyst != null)
                {
                        try {
                                inkeyst.close();
                        } catch (IOException e) {
                                // TODO 自動生成された catch ブロック
                                e.printStackTrace();
                        }
                }

                if(foutst != null)
                {
                        try {
                                foutst.close();
                        } catch (IOException e) {
                                // TODO 自動生成された catch ブロック
                                e.printStackTrace();
                        }
                }

                if(finst != null)
                {
                        try {
                                finst.close();
                        } catch (IOException e) {
                                // TODO 自動生成された catch ブロック
                                e.printStackTrace();
                        }
                }
                return;// 0;
}
else
{    // if the file length is more 1024 bytes
     // read the file a block at a time
     while(rlen > 0 && finst.available()>0)
     {
          // read a block and reduce the remaining byte count
          len = finst.read(cbufb, 1, blen4);
          rlen -= len;
          if((rlen>0) && (len==blen4)){
                                rc=blockDecrypt(cipherI, keyI, cbuf, 8*blen4, pbuf);
                                foutst.write(pbufb, 1, blen4);
                                pfilelen -= blen4;
                        }
          if(rlen<=0){
                                rc=blockDecrypt(cipherI, keyI, cbuf, 8*blen4, pbuf);
                                foutst.write(pbufb, 1, pfilelen);
                                if(inkeyst != null)
                                {
                                        try {
                                                inkeyst.close();
```

```
                                        } catch (IOException e) {
                                                // TODO 自動生成された catch ブロック
                                                e.printStackTrace();
                                        }
                                }

                                if(foutst != null)
                                {
                                        try {
                                                foutst.close();
                                        } catch (IOException e) {
                                                // TODO 自動生成された catch ブロック
                                                e.printStackTrace();
                                        }
                                }

                                if(finst != null)
                                {
                                        try {
                                                finst.close();
                                        } catch (IOException e) {
                                                // TODO 自動生成された catch ブロック
                                                e.printStackTrace();
                                        }
                                }
                                return;
                        }
        }
}
        if(inkeyst != null)
        {
                try {
                        inkeyst.close();
                } catch (IOException e) {
                        // TODO 自動生成された catch ブロック
                        e.printStackTrace();
                }
        }

        if(foutst != null)
        {
                try {
                        foutst.close();
                } catch (IOException e) {
                        // TODO 自動生成された catch ブロック
                        e.printStackTrace();
                }
        }

        if(finst != null)
        {
```

```
                try {
                        finst.close();
                } catch (IOException e) {
                        // TODO 自動生成された catch ブロック
                        e.printStackTrace();
                }
        }

        return;// 0;

}catch (IOException e) {
        // TODO 自動生成された catch ブロック
        e.printStackTrace();
}

}


////////////////////////////////////////////////////////////////////////////
// CmlDC.cpp : コンソール アプリケーション用のエントリ ポイントの定義
//////////////////////////////////////////////////////////////////////////
////////////////////  Cml DC  ////////////////////////////////////////////

        int[] SIGMA = {
                        0xa0,0x9e,0x66,0x7f,0x3b,0xcc,0x90,0x8b,
                        0xb6,0x7a,0xe8,0x58,0x4c,0xaa,0x73,0xb2,
                        0xc6,0xef,0x37,0x2f,0xe9,0x4f,0x82,0xbe,
                        0x54,0xff,0x53,0xa5,0xf1,0xd3,0x6f,0x1c,
                        0x10,0xe5,0x27,0xfa,0xde,0x68,0x2d,0x1d,
                        0xb0,0x56,0x88,0xc2,0xb3,0xe6,0xc1,0xfd};

        int[] KSFT1 = {
                        0,64,0,64,15,79,15,79,30,94,45,109,45,124,60,124,77,13,
                        94,30,94,30,111,47,111,47 };

        int[] KIDX1 = {
                        0,0,4,4,0,0,4,4,4,4,0,0,4,0,4,4,0,0,0,0,4,4,0,0,4,4 };

        int[] KSFT2 = {
                        0,64,0,64,15,79,15,79,30,94,30,94,45,109,45,109,60,124,
                        60,124,60,124,77,13,77,13,94,30,94,30,111,47,111,47 };

        int[] KIDX2 = {
                        0,0,12,12,8,8,4,4,8,8,12,12,0,0,4,4,0,0,8,8,12,12,
                        0,0,4,4,8,8,4,4,0,0,12,12 };

        int[] SBOX = {
          112,130, 44,236,179, 39,192,229,228,133, 87, 53,234, 12,174, 65,
           35,239,107,147, 69, 25,165, 33,237, 14, 79, 78, 29,101,146,189,
          134,184,175,143,124,235, 31,206, 62, 48,220, 95, 94,197, 11, 26,
          166,225, 57,202,213, 71, 93, 61,217,   1, 90,214, 81, 86,108, 77,
```

```
139, 13,154,102,251,204,176, 45,116, 18, 43, 32,240,177,132,153,
223, 76,203,194, 52,126,118,  5,109,183,169, 49,209, 23,  4,215,
 20, 88, 58, 97,222, 27, 17, 28, 50, 15,156, 22, 83, 24,242, 34,
254, 68,207,178,195,181,122,145, 36,  8,232,168, 96,252,105, 80,
170,208,160,125,161,137, 98,151, 84, 91, 30,149,224,255,100,210,
 16,196,  0, 72,163,247,117,219,138,  3,230,218,  9, 63,221,148,
135, 92,131,  2,205, 74,144, 51,115,103,246,243,157,127,191,226,
 82,155,216, 38,200, 55,198, 59,129,150,111, 75, 19,190, 99, 46,
233,121,167,140,159,110,188,142, 41,245,249,182, 47,253,180, 89,
120,152,  6,106,231, 70,113,186,212, 37,171, 66,136,162,141,250,
114,  7,185, 85,248,238,172, 10, 54, 73, 42,104, 60, 56,241,164,
 64, 40,211,123,187,201, 67,193, 21,227,173,244,119,199,128,158 };
```

```
void Camellia_Feistel( int[] x, int xs, int[] k, int ks, int[] y, int ys)//const Byte *x, const Byte *k, Byte *y )
{
    int[] t = new int[8];

    t[0] =  (SBOX[(int) (x[xs+0]^k[ks+0])])&0xff;
    t[1]    =       (SBOX[(int)     (x[xs+1]^k[ks+1])]>>>7^SBOX[(int) (x[xs+1]^k[ks+1])]<<1)&0xff;
    t[2]    =       (SBOX[(int)     (x[xs+2]^k[ks+2])]>>>1^SBOX[(int) (x[xs+2]^k[ks+2])]<<7)&0xff;
    t[3]            =                       (SBOX[(int) (((x[xs+3]^k[ks+3])<<1^(x[xs+3]^k[ks+3])>>>7)&0xff)])&0xff;
    t[4]    =       (SBOX[(int)     (x[xs+4]^k[ks+4])]>>>7^SBOX[(int) (x[xs+4]^k[ks+4])]<<1)&0xff;
    t[5]    =       (SBOX[(int)     (x[xs+5]^k[ks+5])]>>>1^SBOX[(int) (x[xs+5]^k[ks+5])]<<7)&0xff;
    t[6]            =                       (SBOX[(int) (((x[xs+6]^k[ks+6])<<1^(x[xs+6]^k[ks+6])>>>7)&0xff)])&0xff;
    t[7] =  (SBOX[(int) (x[xs+7]^k[ks+7])])&0xff;


    y[(ys+0)] ^= t[0]^t[2]^t[3]^t[5]^t[6]^t[7];
    y[(ys+1)] ^= t[0]^t[1]^t[3]^t[4]^t[6]^t[7];
    y[(ys+2)] ^= t[0]^t[1]^t[2]^t[4]^t[5]^t[7];
    y[(ys+3)] ^= t[1]^t[2]^t[3]^t[4]^t[5]^t[6];
    y[(ys+4)] ^= t[0]^t[1]^t[5]^t[6]^t[7];
    y[(ys+5)] ^= t[1]^t[2]^t[4]^t[6]^t[7];
    y[(ys+6)] ^= t[2]^t[3]^t[4]^t[5]^t[7];
    y[(ys+7)] ^= t[0]^t[3]^t[4]^t[5]^t[6];
}

void Camellia_FLlayer( int[] x, int xs, int[] kl, int kls, int[] kr, int krs)//Byte *x, const Byte *kl, const Byte *kr )
{
    int[] t = new int[4];
    int[] u = new int[4];
```

```
		int[] v = new int[4];

		ByteWord( x, xs, t, 0 );
		ByteWord( kl, kls, u, 0 );
		ByteWord( kr, krs, v, 0 );

		t[1] ^= (((t[0]&u[0])<<1)^((t[0]&u[0])>>>31));
		t[0] ^= (t[1]|u[1]);
		t[2] ^= (t[3]|v[1]);
		t[3] ^= (((t[2]&v[0])<<1)^((t[2]&v[0])>>>31));

		WordByte( t, 0, x, xs );
	}

	void ByteWord( int[] x,int xs, int[] y, int ys)//const Byte *x, Word *y )
	{
		int i;
		for( i=0; i<4; i++ ){
			y[ys+i] = (x[xs+(i<<2)+0]<<24) | (x[xs+(i<<2)+1]<<16)
					| (x[xs+(i<<2)+2]<<8 ) | (x[xs+(i<<2)+3]<<0 );
		}
	}

	void WordByte( int[] x, int xs, int[] y, int ys)//const Word *x, Byte *y )
	{
		int i;
		for( i=0; i<4; i++ ){
			y[ys+(i<<2)+0] = ((x[xs+i]>>>24)&0xff);
			y[ys+(i<<2)+1] = ((x[xs+i]>>>16)&0xff);
			y[ys+(i<<2)+2] = ((x[xs+i]>>> 8)&0xff);
			y[ys+(i<<2)+3] = ((x[xs+i]>>> 0)&0xff);
		}
	}

	void RotBlock( int[] x, int xs, int n, int[] y, int ys)//const Word *x, const int n, Word
*y )
	{
		int r;
		if( (r = (n & 31)) != 0 ){
			y[ys+0]                                                                    =
x[xs+(((n>>>5)+0)&3)]<<r^x[xs+(((n>>>5)+1)&3)]>>>(32-r);
			y[ys+1]                                                                    =
x[xs+(((n>>>5)+1)&3)]<<r^x[xs+(((n>>>5)+2)&3)]>>>(32-r);
		}
		else{
			y[ys+0] = x[xs+(((n>>>5)+0)&3)];
			y[ys+1] = x[xs+(((n>>>5)+1)&3)];
		}
	}

	void SwapHalf( int[] x, int xs)// Byte *x )
```

```
{
	int t;
	int	i;
	for( i=0; i<8; i++ ){
		t = x[xs+i];
		x[xs+i] = x[xs+8+i];
		x[xs+8+i] =	t;
	}
}

void XorBlock( int[] x, int xs, int[] y, int ys, int[] z, int zs)//const Byte *x, const Byte *y,
Byte *z )
{
	int i;
	for( i=0; i<16; i++ ) z[(i+zs)] = (x[(i+xs)] ^ y[(i+ys)]);
}


void Camellia_Ekeygen( int n, int[] k, int[] e)//const int n, const Byte *k, Byte *e )
{
	int[] t = new int[64];
	int[] u = new int[20];
	int	i;

	if( n == 128 ){
		for( i=0 ; i<16; i++ ) t[i] = k[i];
		for( i=16; i<32; i++ ) t[i] = 0;
	}
	else if( n == 192 ){
		for( i=0 ; i<24; i++ ) t[i] = k[i];
		for( i=24; i<32; i++ ) t[i] = (k[i-8]^0xff);
	}
	else if( n == 256 ){
		for( i=0 ; i<32; i++ ) t[i] = k[i];
	}

	XorBlock( t, 0, t, 16, t, 32 );

	Camellia_Feistel( t, 32, SIGMA, 0, t, 40 );
	Camellia_Feistel( t, 40, SIGMA, 8, t, 32 );

	XorBlock( t, 32, t, 0, t, 32 );

	Camellia_Feistel( t, 32, SIGMA, 16, t, 40 );
	Camellia_Feistel( t, 40, SIGMA, 24, t, 32 );

	ByteWord( t, 0,	u, 0 );
	ByteWord( t, 32, u, 4 );

	if( n == 128 ){
		for( i=0; i<26; i+=2 ){
```

```
                        RotBlock( u, KIDX1[i+0], KSFT1[i+0], u, 16 );
                        RotBlock( u, KIDX1[i+1], KSFT1[i+1], u, 18 );
                        WordByte( u, 16, e, i*8 );
                }
        }
        else{
                XorBlock( t, 32, t, 16, t, 48 );

                Camellia_Feistel( t, 48, SIGMA, 32, t, 56 );
                Camellia_Feistel( t, 56, SIGMA, 40, t, 48 );

                ByteWord( t, 16, u, 8   );
                ByteWord( t, 48, u, 12 );

                for( i=0; i<34; i+=2 ){
                        RotBlock( u, KIDX2[i+0], KSFT2[i+0], u, 16 );
                        RotBlock( u, KIDX2[i+1], KSFT2[i+1], u, 18 );
                        WordByte( u, 16, e, (i<<3) );
                }
        }
}


  ////////////////////////////////////////////////////////////////
 ////////////////////////////////////////////////////////////////////////////
void Camellia_Decrypt( int n, int[] c, int cs, int[] e, int es, int[] p, int ps)//const int n,
const Byte *c, const Byte *e, Byte *p )
{
        int i;
        if( n == 128 ){
                XorBlock( c, (cs+0), e, (es+192), p, (ps+0) );
        }
        else{
                XorBlock( c, (cs+0), e, (es+256), p, (ps+0) );

                for( i=2; i>=0; i-- ){
                        Camellia_Feistel( p, (ps+0), e, (es+216+(i<<4)), p, (ps+8) );
                        Camellia_Feistel( p, (ps+8), e, (es+208+(i<<4)), p, (ps+0) );
                }

                Camellia_FLlayer( p, (ps+0), e, (es+200), e, (es+192) );
        }

        for( i=2; i>=0; i-- ){
                Camellia_Feistel( p, (ps+0), e, (es+152+(i<<4)), p, (ps+8) );
                Camellia_Feistel( p, (ps+8), e, (es+144+(i<<4)), p, (ps+0) );
        }

        Camellia_FLlayer( p, (ps+0), e, (es+136), e, (es+128) );

        for( i=2; i>=0; i-- ){
```

```
                Camellia_Feistel( p, (ps+0), e, (es+88+(i<<4)), p, (ps+8) );
                Camellia_Feistel( p, (ps+8), e, (es+80+(i<<4)), p, (ps+0) );
            }

            Camellia_FLlayer( p, (ps+0), e, (es+72), e, (es+64) );

            for( i=2; i>=0; i-- ){
                Camellia_Feistel( p, (ps+0), e, (es+24+(i<<4)), p, (ps+8) );
                Camellia_Feistel( p, (ps+8), e, (es+16+(i<<4)), p, (ps+0) );
            }

            SwapHalf( p , ps);
            XorBlock( p, (ps+0), e, (es+0), p, (ps+0) );
        }


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

        // CmlEC.cpp： コンソール アプリケーション用のエントリ ポイントの定義

        // 暗号文の HEX 表示用
        char toChar( int c )
        {
            if( c >= 0 && c <= 9 )                          // 0～9 ならば
                return (char)( c + 0x30 );  // ASCII に変換して返す
            else if( c >= 10 && c <= 15 )          // 10～15 ならば
                return (char)( c + 0x37 );  // A～F の ASCII を返す
            else
                return ' ';
        }


////////////////////////////////////////////////////////////////////////////////
// CmlDC.cpp： コンソール アプリケーション用のエントリ ポイントの定義 ////////////////////////
    int CmlDC(String keyfn, String ctfn, String ptfn)              // 引数へのポインタ
    {
        File fkey;
        int i,len;
        byte[] c_klen = new byte[5];
        int[] pass1 = new int[64];
        byte[] pass2b = new byte[128];
        int j,k;
        int[] exkey = new int[512];
            ////////////////////////////////
        int block;
        int[] bufp;
        int[] bufc;
        byte[] bufbp;
        byte[] bufbc;
        int mesLength; // 平文長（バイト）
        int lenp = 0;
```

```
File fin = new File(ctfn);
fin.getParentFile().mkdir();
FileInputStream finst=null;
try {
	finst = new FileInputStream(fin);
} catch (FileNotFoundException e5) {
	// TODO 自動生成された catch ブロック
	e5.printStackTrace();
}

File fout = new File(ptfn);
fout.getParentFile().mkdir();
FileOutputStream foutst=null;
try {
	foutst = new FileOutputStream(fout);
} catch (FileNotFoundException e5) {
	// TODO 自動生成された catch ブロック
	e5.printStackTrace();
}

fkey = new File(keyfn);
fkey.getParentFile().mkdir();
FileInputStream inkeyst=null;
try {
	inkeyst = new FileInputStream(fkey);
	inkeyst.read(c_klen);
	inkeyst.read(pass2b);
} catch (IOException e3) {
	// TODO 自動生成された catch ブロック
	e3.printStackTrace();
}//127

len = atoi(c_klen);

for(i=0; i<len/8; i++){
	j = pass2b[2*i];
	k = pass2b[2*i+1];
	if(j>=0x30 && j<=0x39) j = (j-0x30);
	else{
		if(j>=0x41 && j<=0x46) j = (j-0x41+0x0A);
	}
	if(k>=0x30 && k<=0x39) k = (k-0x30);
	else{
		if(k>=0x41 && k<=0x46) k = (k-0x41+0x0A);
	}
	pass1[i] = j*0x10 + k ;
}
pass1[(int) (len/8)] = 0;
```

```
Camellia_Ekeygen( len, pass1,   exkey );

// 暗号文
try {
        int filelen = finst.available();

        int head = 4;
        mesLength = filelen;

        if(mesLength <= 1024){
                int rd = 0;
                if(mesLength%16 != 0){ rd = 1;}
                else{ rd = 0;}
                block = (int) (mesLength/16 + rd);
                bufp = new int[block*16 + 2];
                bufc = new int[block*16 + 2];
                bufbp = new byte[block*16 + 2];
                bufbc = new byte[block*16 + 2];

                // 暗文
                try {
                        int rl = finst.read(bufbc, 0, mesLength);
                        for( i =0; i<rl; i++)
                        {
                                bufc[i] = bufbc[i];
                                if(bufc[i]<0){
                                        int ll = bufbc[i];
                                        if(ll<0){
                                                bufbc[i] ^= 0x80;
                                                ll = bufbc[i];
                                                ll += 0x80;
                                        }
                                        bufc[i] = ll;
                                }
                        }
                } catch (IOException e1) {
                        // TODO 自動生成された catch ブロック
                        e1.printStackTrace();
                }

                // 復号化実行
                for(i=0;i<block;i++){
                        Camellia_Decrypt( len,   bufc,  (i*16),  exkey,  0,
bufp, (i*16));
                //n=鍵長      bufc=暗号文      exkey=拡張鍵      bufp=平文
                }

                for(i=0; i<block*16 ; i++){
                        bufbp[i] = (byte)bufp[i];
                }
```

```
                // 復号文を書き込む
                byte[] tmpch4 = new byte[4];
                tmpch4[0] = bufbp[0];
                tmpch4[1] = bufbp[1];
                tmpch4[2] = bufbp[2];
                tmpch4[3] = bufbp[3];
                int jj = 0;
                int tmp = 0;
                for (int p = 0; p < tmpch4.length; p++) {
                        tmp = (int)(tmpch4[3-p] & 0xff);
                        jj = (jj << 8) | tmp;
                }
                lenp = jj;


                bufbp[lenp+head] = 0;
                try {
                        foutst.write(bufbp, head, lenp);
                } catch (IOException e) {
                        // TODO 自動生成された catch ブロック
                        e.printStackTrace();
                }
        }
        else{
                int rd = 0;
                if(mesLength%16 != 0){ rd = 1;}
                else{ rd = 0;}
                block = (int) (mesLength/16 + rd);
                bufp = new int[1024 + 2];
                bufc = new int[1024 + 2];
                bufbp = new byte[1024 + 2];
                bufbc = new byte[1024 + 2];

                int rBlen = block;
                int r = 0;
                do{
                        // 暗文
                        try {
                                int rl = finst.read(bufbc, 0, 1024);
                                for(i =0; i<1024; i++){
                                        bufc[i] = bufbc[i];
                                }
                        } catch (IOException e1) {
                                // TODO 自動生成された catch ブロック
                                e1.printStackTrace();
                        }

                        if(rBlen >= 1024/16){ block = 1024/16; }
                        if(rBlen < 1024/16){ block = (int) rBlen;}

                        // 復号化実行
```

```
for(i=0;i<block;i++){
        Camellia_Decrypt( len, bufc, i*16, exkey,
0, bufp, i*16 );
}

for(i=0; i<block*16 ; i++){
        bufbp[i] = (byte)bufp[i];
}

// 復号文を書き込む
if(r==0){
        byte[] tmpch4 = new byte[4];
        tmpch4[0] = bufbp[0];
        tmpch4[1] = bufbp[1];
        tmpch4[2] = bufbp[2];
        tmpch4[3] = bufbp[3];
        int tmp =0, jj = 0;
        for (int p = 0; p < tmpch4.length; p++) {
                tmp = (int)(tmpch4[3-p] & 0xff);
                jj = (jj << 8) | tmp;
        }
        lenp = jj;


        try {
                foutst.write(bufbp,          head,
1024-head);
        } catch (IOException e) {
                // TODO 自動生成された catch
ブロック
                e.printStackTrace();
        }

        lenp -= 1024-head;
}
else{
        if(rBlen >= 1024/16){
                try {
                        for(i=0; i<1024; i++){
                                bufbp[i]      =
(byte)bufp[i];
                        }
                        foutst.write(bufbp,    0,
1024);
                } catch (IOException e) {
                        // TODO 自動生成され
た catch ブロック
                        e.printStackTrace();
                }
                lenp -= 1024;
        }
```

```
                                        else{
                                                try {
                                                        for(i=0; i<lenp; i++){
                                                                bufbp[i]        =
(byte)bufp[i];
                                                        }
                                                        foutst.write(bufbp,     0,
lenp);
                                                } catch (IOException e) {
                                                        // TODO 自動生成され
た catch ブロック
                                                        e.printStackTrace();
                                                }
                                        }
                                }
                                r += 1;
                                rBlen -= 1024/16;
                        }while(rBlen>0);
                }

                if(inkeyst != null)
                {
                        try {
                                inkeyst.close();
                        } catch (IOException e) {
                                // TODO 自動生成された catch ブロック
                                e.printStackTrace();
                        }
                }

                if(foutst != null)
                {
                        try {
                                foutst.close();
                        } catch (IOException e) {
                                // TODO 自動生成された catch ブロック
                                e.printStackTrace();
                        }
                }

                if(finst != null)
                {
                        try {
                                finst.close();
                        } catch (IOException e) {
                                // TODO 自動生成された catch ブロック
                                e.printStackTrace();
                        }
                }
        } catch (IOException e2) {
                // TODO 自動生成された catch ブロック
```

```
                                e2.printStackTrace();
                        }

                return 0;
        }




    ////////////////////////////////////////////////////////////////
    //////////////////    AES DC    ////////////////////////////////////////

    int BC, KC, ROUNDS;
//    int s;

    int[] Logtable = {
                0,    0, 25,    1, 50,    2, 26,198, 75,199, 27,104, 51,238,223,    3,
              100,    4,224, 14, 52,141,129,239, 76,113,    8,200,248,105, 28,193,
              125,194, 29,181,249,185, 39,106, 77,228,166,114,154,201,    9,120,
              101, 47,138,    5, 33, 15,225, 36, 18,240,130, 69, 53,147,218,142,
              150,143,219,189, 54,208,206,148, 19, 92,210,241, 64, 70,131, 56,
              102,221,253, 48,191,    6,139, 98,179, 37,226,152, 34,136,145, 16,
              126,110, 72,195,163,182, 30, 66, 58,107, 40, 84,250,133, 61,186,
               43,121, 10, 21,155,159, 94,202, 78,212,172,229,243,115,167, 87,
              175, 88,168, 80,244,234,214,116, 79,174,233,213,231,230,173,232,
               44,215,117,122,235, 22, 11,245, 89,203, 95,176,156,169, 81,160,
              127, 12,246,111, 23,196, 73,236,216, 67, 31, 45,164,118,123,183,
              204,187, 62, 90,251, 96,177,134, 59, 82,161,108,170, 85, 41,157,
              151,178,135,144, 97,190,220,252,188,149,207,205, 55, 63, 91,209,
               83, 57,132, 60, 65,162,109, 71, 20, 42,158, 93, 86,242,211,171,
               68, 17,146,217, 35, 32, 46,137,180,124,184, 38,119,153,227,165,
              103, 74,237,222,197, 49,254, 24, 13, 99,140,128,192,247,112,    7};

    int[] Alogtable = {
                1,    3,    5, 15, 17, 51, 85,255, 26, 46,114,150,161,248, 19, 53,
               95,225, 56, 72,216,115,149,164,247,    2,    6, 10, 30, 34,102,170,
              229, 52, 92,228, 55, 89,235, 38,106,190,217,112,144,171,230, 49,
               83,245,    4, 12, 20, 60, 68,204, 79,209,104,184,211,110,178,205,
               76,212,103,169,224, 59, 77,215, 98,166,241,    8, 24, 40,120,136,
              131,158,185,208,107,189,220,127,129,152,179,206, 73,219,118,154,
              181,196, 87,249, 16, 48, 80,240, 11, 29, 39,105,187,214, 97,163,
              254, 25, 43,125,135,146,173,236, 47,113,147,174,233, 32, 96,160,
              251, 22, 58, 78,210,109,183,194, 93,231, 50, 86,250, 21, 63, 65,
              195, 94,226, 61, 71,201, 64,192, 91,237, 44,116,156,191,218,117,
              159,186,213,100,172,239, 42,126,130,157,188,223,122,142,137,128,
              155,182,193, 88,232, 35,101,175,234, 37,111,177,200, 67,197, 84,
              252, 31, 33, 99,165,244,    7,    9, 27, 45,119,153,176,203, 70,202,
               69,207, 74,222,121,139,134,145,168,227, 62, 66,198, 81,243, 14,
               18, 54, 90,238, 41,123,141,140,143,138,133,148,167,242, 13, 23,
               57, 75,221,124,132,151,162,253, 28, 36,108,180,199, 82,246,    1};
```

```
int[] S = {
        99,124,119,123,242,107,111,197, 48,   1,103, 43,254,215,171,118,
       202,130,201,125,250, 89, 71,240,173,212,162,175,156,164,114,192,
       183,253,147, 38, 54, 63,247,204, 52,165,229,241,113,216, 49, 21,
         4,199, 35,195, 24,150,   5,154,   7, 18,128,226,235, 39,178,117,
         9,131, 44, 26, 27,110, 90,160, 82, 59,214,179, 41,227, 47,132,
        83,209,   0,237, 32,252,177, 91,106,203,190, 57, 74, 76, 88,207,
       208,239,170,251, 67, 77, 51,133, 69,249,   2,127, 80, 60,159,168,
        81,163, 64,143,146,157, 56,245,188,182,218, 33, 16,255,243,210,
       205, 12, 19,236, 95,151, 68, 23,196,167,126, 61,100, 93, 25,115,
        96,129, 79,220, 34, 42,144,136, 70,238,184, 20,222, 94, 11,219,
       224, 50, 58, 10, 73,   6, 36, 92,194,211,172, 98,145,149,228,121,
       231,200, 55,109,141,213, 78,169,108, 86,244,234,101,122,174,   8,
       186,120, 37, 46, 28,166,180,198,232,221,116, 31, 75,189,139,138,
       112, 62,181,102, 72,   3,246, 14, 97, 53, 87,185,134,193, 29,158,
       225,248,152, 17,105,217,142,148,155, 30,135,233,206, 85, 40,223,
       140,161,137, 13,191,230, 66,104, 65,153, 45, 15,176, 84,187, 22};

int[] Si = {
        82,   9,106,213, 48, 54,165, 56,191, 64,163,158,129,243,215,251,
       124,227, 57,130,155, 47,255,135, 52,142, 67, 68,196,222,233,203,
        84,123,148, 50,166,194, 35, 61,238, 76,149, 11, 66,250,195, 78,
         8, 46,161,102, 40,217, 36,178,118, 91,162, 73,109,139,209, 37,
       114,248,246,100,134,104,152, 22,212,164, 92,204, 93,101,182,146,
       108,112, 72, 80,253,237,185,218, 94, 21, 70, 87,167,141,157,132,
       144,216,171,   0,140,188,211, 10,247,228, 88,   5,184,179, 69,   6,
       208, 44, 30,143,202, 63, 15,   2,193,175,189,   3,   1, 19,138,107,
        58,145, 17, 65, 79,103,220,234,151,242,207,206,240,180,230,115,
       150,172,116, 34,231,173, 53,133,226,249, 55,232, 28,117,223,110,
        71,241, 26,113, 29, 41,197,137,111,183, 98, 14,170, 24,190, 27,
       252, 86, 62, 75,198,210,121, 32,154,219,192,254,120,205, 90,244,
        31,221,168, 51,136,   7,199, 49,177, 18, 16, 89, 39,128,236, 95,
        96, 81,127,169, 25,181, 74, 13, 45,229,122,159,147,201,156,239,
       160,224, 59, 77,174, 42,245,176,200,235,187, 60,131, 83,153, 97,
        23, 43,   4,126,186,119,214, 38,225,105, 20, 99, 85, 33, 12,125};


int[] RC = {
        0x00,0x01,0x02,0x04,0x08,0x10,0x20,0x40,0x80,
        0x1b,0x36,0x6c,0xd8,0xab,0x4d,0x9a,0x2f,0x5e,
        0xbc,0x63,0xc6,0x97,0x35,0x6a,0xd4,0xb3,0x7d,
        0xfa,0xef,0xc5};



static int[][] shifts = {
        {0,1,2,3},
        {0,1,2,3},
        {0,1,2,3},
        {0,1,2,4},
```

```
        {0,1,3,4}};

static int[][] numrounds = {
        {10,11,12,13,14},
        {11,11,12,13,14},
        {12,12,12,13,14},
        {13,13,13,13,14},
        {14,14,14,14,14}};



int mul(int a, int b) {
        if(a!=0 && b!=0) return Alogtable[(Logtable[a] + Logtable[b])%255];
        else return 0;
}

void AddRoundKey(int[][] a, int[][] rk) {
        int i, j;

        for(i = 0; i<4; i++)
                for(j=0;j<BC;j++) a[i][j] ^= rk[i][j];
}

void SubBytes(int[][] a, int[] box){
        int i,j;

        for(i=0;i<4;i++)
                for(j=0;j<BC;j++) a[i][j] = box[a[i][j]] ;
}

void ShiftRows(int[][] a, int d){
        int[] tmp = new int[8];
        int i,j;

        if(d==0){
                for(i=1;i<4;i++){
                        for(j=0;j<BC;j++)
                                tmp[j] = a[i][(j+shifts[BC-4][i]) % BC];
                        for(j=0;j<BC;j++) a[i][j] = tmp[j];
                }
        }
        else{
                for(i=1;i<4;i++){
                        for(j=0;j<BC;j++)
                                tmp[j] = a[i][(BC+j-shifts[BC-4][i]) % BC];
                        for(j=0;j<BC;j++) a[i][j] = tmp[j];
                }
        }
}

void MixColumns(int[][] a){
```

```
	int[][] b = new int[4][8];
	int i,j;

	for(j=0;j<BC;j++)
		for(i=0;i<4;i++)
			b[i][j] = mul(2,a[i][j])
				^ mul(3,a[(i+1) % 4][j])
				^ a[(i+2) % 4][j]
				^ a[(i+3) % 4][j];
	for(i=0;i<4;i++)
		for(j=0;j<BC;j++) a[i][j] = b[i][j];
}

void InvMixColumns(int[][] a){
	int[][] b = new int[4][8];
	int i,j;

	for(j=0;j<BC;j++)
	for(i=0;i<4;i++)
		b[i][j] = mul(0xe, a[i][j])
			^ mul(0xb, a[(i+1) % 4][j])
			^ mul(0xd, a[(i+2) % 4][j])
			^ mul(0x9, a[(i+3) % 4][j]);
	for(i=0;i<4;i++)
		for(j=0;j<BC;j++) a[i][j] = b[i][j];
}

int KeyExpansion(int[][] k, int[][][] W){
	int i,j,t,RCpointer = 1;
	int[][] tk = new int[4][8];

	for(j=0;j<KC;j++)
		for(i=0;i<4;i++)
			tk[i][j] = k[i][j];
	t = 0;

	for(j=0;(j<KC) && (t<(ROUNDS+1)*BC); j++, t++)
		for(i=0;i<4;i++) W[t / BC][i][t % BC] = tk[i][j];

	while(t<(ROUNDS+1)*BC){
		for(i=0;i<4;i++)
			tk[i][0] ^= S[tk[(i+1)%4][KC-1]];
		tk[0][0] ^= RC[RCpointer++];

		if(KC<=6)
			for(j=1;j<KC;j++)
				for(i=0;i<4;i++) tk[i][j] ^= tk[i][j-1];
		else{
			for(j=1;j<4;j++)
				for(i=0;i<4;i++) tk[i][j] ^= tk[i][j-1];
			for(i=0;i<4;i++) tk[i][4] ^= S[tk[i][3]];
```

```
                        for(j=5;j<KC;j++)
                                for(i=0;i<4;i++) tk[i][j] ^= tk[i][j-1];
                }
                for(j=0; (j<KC) && (t<(ROUNDS+1)*BC); j++, t++)
                        for(i=0;i<4;i++) W[t/BC][i][t%BC] = tk[i][j];
        }
        return 0;
  }

  int Decrypt(int[][] a, int[][][] rk){
        int r;

        AddRoundKey(a,rk[ROUNDS]);
        SubBytes(a,Si);
        ShiftRows(a,1);

        for(r=ROUNDS-1;r>0;r--){
                AddRoundKey(a,rk[r]);
                InvMixColumns(a);
                SubBytes(a,Si);
                ShiftRows(a,1);
        }
        AddRoundKey(a,rk[0]);

        return 0;
  }


////////////////////////////////////////////////////////////////////////////
  int atoi( byte s[] ) {
            int i, n, sign;

            for( i = 0; s[i] == ' ' ; i++ )   //先頭の空白を読み飛ばす
                ;
            sign = ( s[i] == '-' ) ? -1 : 1;      //符号を保存する
            if( s[i] == '-' || s[i] == '+' )      //符号を飛ばす
                i++;
            for( n = 0; i< s.length - 2 ; i++ )   //s[i]が数字のあいだ、n へ
                n = 10 * n + ( s[i] - '0' );
            return sign * n;                              //符号を反映
        }

  ////////////////////////////////////////////////////////////////
  ////////////////////////////////////////////////////////////////
  int aesdc(String keyfn, String ctfn, String ptfn)
  {
        int i,j,klen,blen;
        int[][] a = new int[4][8];
        int[][][] rk = new int[14+1][4][8];
        int[][] sk = new int[4][8];
        byte[] c_klen = new byte[5];
```

```
        byte[] c_blen = new byte[5];
        byte[] pass = new byte[127];
        byte[] dbuf         = new byte[64];
    int len=0, rlen = 0;
        int blen4;

  ////////////////////////////////////////////////////////
        try {

//                  FileOutputStream foutst = openFileOutput(ptfn,MODE_PRIVATE);
//                  FileInputStream finst = openFileInput(ctfn);

                    File fin = new File(ctfn);
                    fin.getParentFile().mkdir();
                    FileInputStream finst=null;
                    try {
                            finst = new FileInputStream(fin);
                    } catch (FileNotFoundException e5) {
                            // TODO 自動生成された catch ブロック
                            e5.printStackTrace();
                    }

                    File fout = new File(ptfn);
                    fout.getParentFile().mkdir();
                    FileOutputStream foutst=null;
                    try {
                            foutst = new FileOutputStream(fout);
                    } catch (FileNotFoundException e5) {
                            // TODO 自動生成された catch ブロック
                            e5.printStackTrace();
                    }

                    File fkey = new File(keyfn);
                    fkey.getParentFile().mkdir();
                    FileInputStream inkeyst=null;
                    try {
                            inkeyst = new FileInputStream(fkey);
                    } catch (FileNotFoundException e5) {
                            // TODO 自動生成された catch ブロック
                            e5.printStackTrace();
                    }

                    inkeyst.read(c_klen);// 2 5 6 CR LF          5byte
                    inkeyst.read(c_blen);// 2 5 6 CR LF          5byte
                    inkeyst.read(pass);//127

                    klen = atoi(c_klen)/32;
                    blen = atoi(c_blen)/32;
                    blen4 = (int)blen*4;

                    KC = klen;
```

```
		if(KC<4 || 8<KC){
			return (-1);
		}

		BC = blen;
		if(BC<4 || 8<BC){
			return (-1);
		}

		ROUNDS = numrounds[KC-4][BC-4];

		char cl,cr;
		int k = 0;
		for(j=0;j<KC;j++){
			for(i=0;i<4;i++){
				if(pass[k]>=0x30 && pass[k]<=0x39){cl = (char) (pass[k]-0x30);}
				else if(pass[k]>=0x41 && pass[k]<=0x46){cl = (char) (pass[k]-0x37);}
				else if(pass[k]>=0x61 && pass[k]<=0x66){cl = (char) (pass[k]-0x57);}
				else cl = 0;
				if(pass[k+1]>=0x30 && pass[k+1]<=0x39){cr = (char)
(pass[k+1]-0x30);}
				else if(pass[k+1]>=0x41 && pass[k+1]<=0x46){cr = (char)
(pass[k+1]-0x37);}
				else if(pass[k+1]>=0x61 && pass[k+1]<=0x66){cr = (char)
(pass[k+1]-0x57);}
				else cr = 0;
				sk[i][j] = ((cl<<4)|(cr));
				k += 2;
			}
		}

		KeyExpansion(sk,rk);

		s = 4;//sizeof(unsigned int);
		rlen = finst.available();
		len = finst.read(dbuf, 0, blen4);
		rlen -= len;

		// decrypt the top 16 bytes of the buffer
		k=0;
		for(j=0;j<BC;j++){
			for(i=0;i<4;i++){
				int ll = dbuf[k];
				if(ll<0){
					dbuf[k] ^= 0x80;
					ll = dbuf[k];
					ll += 0x80;
				}
				a[i][j] = ll;
				k++;
			}
```

```
}

Decrypt(a,rk);

// write the IV and the encrypted file bytes
k=0;
for(j=0;j<BC;j++){
        for(i=0;i<4;i++){
                dbuf[k] = (byte) a[i][j];
                k++;
        }
}

int wlen;// = *((unsigned int*)dbuf);
int jj = 0, tmp;
byte[] tmpch4 = new byte[4];
tmpch4[0] = dbuf[0];
tmpch4[1] = dbuf[1];
tmpch4[2] = dbuf[2];
tmpch4[3] = dbuf[3];
for (int p = 0; p < tmpch4.length; p++) {
        tmp = (int)(tmpch4[3-p] & 0xff);
        jj = (jj << 8) | tmp;
}
wlen = jj;

if(wlen <= blen4-s){
        try {
                foutst.write(dbuf, s, wlen);
        } catch (IOException e) {
                // TODO 自動生成された catch ブロック
                e.printStackTrace();
        }
}
else{
        wlen -= blen4-s;
        try {
                foutst.write(dbuf, s, (int) (blen4-s));
        } catch (IOException e) {
                // TODO 自動生成された catch ブロック
                e.printStackTrace();
        }
}

if( rlen <= blen4 && rlen>0 )
{    // if the original file length is less than or equal to 16 bytes
        // read the bytes of the file and verify length
        try {
                len = finst.read(dbuf, 0, blen4);
                rlen -= len;
        } catch (IOException e) {
```

```
                // TODO 自動生成された catch ブロック
                e.printStackTrace();
            }

            k=0;
            for(j=0;j<BC;j++){
                for(i=0;i<4;i++){
                    int ll = dbuf[k];
                    if(ll<0){
                        dbuf[k] ^= 0x80;
                        ll = dbuf[k];
                        ll += 0x80;
                    }
                    a[i][j] = ll;
                    k++;
                }
            }

            Decrypt(a,rk);

            k=0;
            for(j=0;j<BC;j++){
                for(i=0;i<4;i++){
                    dbuf[k] = (byte) a[i][j];
                    k++;
                }
            }
            try {
                foutst.write(dbuf, 0, wlen);
            } catch (IOException e) {
                // TODO 自動生成された catch ブロック
                e.printStackTrace();
            }
        }
        else
        {
            int fab;

            fab = finst.available();

            while( rlen>0 && fab>0)
            {
                // input a block and reduce the remaining byte count
                try {
                    len = finst.read(dbuf, 0, blen4);
                    rlen -= len;
                } catch (IOException e1) {
                    // TODO 自動生成された catch ブロック
                    e1.printStackTrace();
                }
                if(len>0){
```

```
		try {
			fab = finst.available();
		} catch (IOException e) {
			// TODO 自動生成された catch ブロック
			e.printStackTrace();
		}
}

// verify the length of the read operation
if(len != blen4 )
		return -1;

// decrypt input buffer
k=0;
for(j=0;j<BC;j++){
		for(i=0;i<4;i++){
			int ll = dbuf[k];
			if(ll<0){
				dbuf[k] ^= 0x80;
				ll = dbuf[k];
				ll += 0x80;
			}
			a[i][j] = ll;
			k++;
		}
}

Decrypt(a,rk);

k=0;
for(j=0;j<BC;j++){
		for(i=0;i<4;i++){
			dbuf[k] = (byte) a[i][j];
			k++;
		}
}

if(wlen < blen4){
		try {
			foutst.write(dbuf, 0, wlen);
		} catch (IOException e) {
			// TODO 自動生成された catch ブロック
			e.printStackTrace();
		}
		break;
}
else{
		try {
			foutst.write(dbuf, 0, blen4);
		} catch (IOException e) {
			// TODO 自動生成された catch ブロック
```

```
                                        e.printStackTrace();
                                }
                        }
                        if(wlen > blen4){wlen -= blen4;}
                }
        }


        if(inkeyst != null)
        {
                try {
                        inkeyst.close();
                } catch (IOException e) {
                        // TODO 自動生成された catch ブロック
                        e.printStackTrace();
                }
        }

        if(foutst != null)
        {
                try {
                        foutst.close();
                } catch (IOException e) {
                        // TODO 自動生成された catch ブロック
                        e.printStackTrace();
                }
        }

        if(finst != null)
        {
                try {
                        finst.close();
                } catch (IOException e) {
                        // TODO 自動生成された catch ブロック
                        e.printStackTrace();
                }
        }
    }catch (FileNotFoundException e2) {
        // TODO 自動生成された catch ブロック
        e2.printStackTrace();
    } catch (IOException e2) {
        // TODO 自動生成された catch ブロック
        e2.printStackTrace();
    }

    return 0;
}



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

```
////////////////////// Bmp56DC ////////////////////////////////////////////////////////
public void bmp56dc(String keyfn,   String srcfn, String dstfn)   // 復号化
{
        int nsize, fsize;
        int j, k, jj;
        int i, mn;
        byte tmpch4[] = new byte[4];
        byte tmprbuf2[] = new byte[2];
        byte tmprbuf1[] = new byte[1];
        byte key[] = new byte[64];//32];
        byte bmpHeader54[] = new byte[54];
        int q;
        int tmp;
////////////////////////////////////// for 56 version
        key[0] = 0x31;
        key[1] = 0x32;
        key[2] = 0x33;
        key[3] = 0x34;
        key[4] = 0x35;
        key[5] = 0x36;
        key[6] = 0x37;

        mn = 7;
////////////////////////////////////////////////

        try {
//              FileOutputStream writer2 = openFileOutput(dstfn,MODE_PRIVATE);
//              FileInputStream in2 = openFileInput(srcfn);

//              FileOutputStream foutst = openFileOutput(ptfn,MODE_PRIVATE);
//              FileInputStream finst = openFileInput(ctfn);

                File fin = new File(srcfn);
                fin.getParentFile().mkdir();
                FileInputStream in2=null;
                try {
                        in2 = new FileInputStream(fin);
                } catch (FileNotFoundException e5) {
                        // TODO 自動生成された catch ブロック
                        e5.printStackTrace();
                }

                File fout = new File(dstfn);
                fout.getParentFile().mkdir();
                FileOutputStream writer2=null;
                try {
                        writer2 = new FileOutputStream(fout);
                } catch (FileNotFoundException e5) {
                        // TODO 自動生成された catch ブロック
                        e5.printStackTrace();
                }
```

```
if(in2.available() > 0){
            in2.read(bmpHeader54);
}

if(in2.available() > 0){
            in2.read(tmpch4);
}
jj = 0;
for (int p = 0; p < tmpch4.length; p++) {
            tmp = (int)(tmpch4[3-p] & 0xff);
      jj = (jj << 8) | tmp;
}
j = jj^(jj>>>16);
nsize = j & 0x0000ffff;

if(in2.available() > 0){
            in2.read(tmpch4);
}
jj = 0;
for (int p = 0; p < tmpch4.length; p++) {
            tmp = (int)(tmpch4[3-p] & 0xff);
      jj = (jj << 8) | tmp;
}
j = jj^(jj>>>16);
k = j & 0x0000ffff;
nsize = nsize + (k<<16);

if(in2.available() > 0){
            in2.read(tmpch4);
}
jj = 0;
for (int p = 0; p < tmpch4.length; p++) {
      jj = (jj << 8) | (tmpch4[3-p] & 0xff);
}
j = jj^(jj>>>16);
fsize = j & 0x0000ffff;

if(in2.available() > 0){
            in2.read(tmpch4);
}
jj = 0;
for (int p = 0; p < tmpch4.length; p++) {
      jj = (jj << 8) | (tmpch4[3-p] & 0xff);
}
j = jj^(jj>>>16);
k = j & 0x0000ffff;
fsize = fsize + (k<<16);

byte pfName[] = new byte[nsize+8];
```

```
for(i=0; i<nsize/2 ; i++){
		if(in2.available() > 0){
			in2.read(tmpch4);
		}
		jj = 0;
		for (int p = 0; p < tmpch4.length; p++) {
		     jj = (jj << 8) | (tmpch4[3-p] & 0xff);
		}
		j = jj^(jj>>16);
		k = j & 0x0000ffff;
		pfName[2*i] = (byte)(k>>8);
		pfName[2*i+1] = (byte)(k&0x000000ff);
}
for(i=nsize/2; i<(nsize+5)/2 ; i++){
		pfName[2*i] = 0;//(Byte) null;
		pfName[2*i+1] = 0;//(Byte) null;
}
if(nsize%2 == 0){
		if(in2.available() > 0){
			in2.read(tmpch4);
		}
}
if(nsize%2 == 1){
		if(in2.available() > 0){
			in2.read(tmpch4);
		}
		jj = 0;
		for (int p = 0; p < tmpch4.length; p++) {
		     jj = (jj << 8) | (tmpch4[3-p] & 0xff);
		}
		j = jj^(jj>>16);
		k = j & 0x0000ffff;
		pfName[nsize-1] = (byte)(k>>8);
}



for(i=0 ; i<fsize ; i+=2){
		q = fsize - i - 2;
		if(q >= 0){
			if(4 == in2.read(tmpch4)){
				jj = 0;
				for (int p = 0; p < tmpch4.length; p++) {
				     jj = (jj << 8) | (tmpch4[3-p] & 0xff);
				}
				j = jj^(jj>>16);
				k = j & 0x0000ffff;
				tmprbuf2[0] = (byte)(k>>8);
				tmprbuf2[0] ^= (byte)key[(i/2)%mn];
				tmprbuf2[1] = (byte)(k&0x000000ff);
```

```
                                        tmprbuf2[1] ^= (byte)key[(i/2)%mn];
                                        writer2.write(tmprbuf2);
                                }
                        }
                        if(q < 0){ // q==-1
                                if(4 == in2.read(tmpch4)){
                                        jj = 0;
                                        for (int p = 0; p < tmpch4.length; p++) {
                                            jj = (jj << 8) | (tmpch4[3-p] & 0xff);
                                        }
                                        j = jj^(jj>>16);
                                        k = j & 0x0000ffff;
                                        tmprbuf1[0] = (byte)(k>>8);
                                        tmprbuf1[0] ^= (byte)key[(i/2)%mn];
                                        writer2.write(tmprbuf1);
                                }
                        }
                }

                writer2.flush();

        if (writer2 != null)
                        writer2.close();
                if (in2 != null)
                        in2.close();

        } catch (FileNotFoundException e) {
        e.printStackTrace();
    }catch (IOException e) {
                System.out.println("添付ファイルの保存に失敗しました。" + e);
        } finally {

        }



}


public static void copyTransfer(String srcPath, String destPath)
    throws IOException {

    FileChannel srcChannel = new
        FileInputStream(srcPath).getChannel();
    FileChannel destChannel = new
        FileOutputStream(destPath).getChannel();
    try {
        srcChannel.transferTo(0, srcChannel.size(), destChannel);
    } finally {
        srcChannel.close();
        destChannel.close();
```

```
    }
}

private void cancelreturn() {
    finish();
}

public void mailviewattach() {//指定された添付ファイルの復号化、保存
        MailViewAttachData mailviewattachData = new MailViewAttachData(idvaf , idtxtaf,
fnamelist, fpathlist, fname, fpath);

        Intent i = new Intent(this, yu.com.pcs.jp.sumaho.cg5mail.MailViewAttachActivity.class);
        i.putExtra("mailviewattachData", mailviewattachData);
        this.startActivityForResult(i, 1);
}


public void returnmail() {
        MailData mailData2 = new MailData(idv , idtxt, attach, subject, addressfrom,
                        addressto, date, size , priority , read, state, messagenum, flag, xmailer,
alldata);

        Intent i = new Intent(this, yu.com.pcs.jp.sumaho.cg5mail.MailRetEditActivity.class);
        i.putExtra("mailData2", mailData2);
        this.startActivityForResult(i, 2);
}


@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
  super.onActivityResult(requestCode, resultCode, data);

        int ret = 0;
        int encrypt = 0;
        int oi_ingroup = 0;
        String oc_encsoft = "";
        String oc_enckey = "";
        String buf2 = addressfrom;
        String buf3 = "";
        String dcp1 = dcprg1;
        String dck1 = getMount_sd() + "/" +dckey1;
        String dcp2 = dcprg2;
        String dck2 = getMount_sd() + "/" +dckey2;
        String dcp3 = dcprg3;
        String dck3 = getMount_sd() + "/" +dckey3;
        String dcp4 = dcprg4;
        String dck4 = getMount_sd() + "/" +dckey4;
        String dcp5 = dcprg5;
        String dck5 = getMount_sd() + "/" +dckey5;

        int iprg1 = 0;
```

```
        String kf1 = "";
        String pt1 = "";
        String ct1 = "";
        FileInputStream in5 = null;
        FileOutputStream writer5 = null;

if (requestCode == 1 && resultCode == RESULT_OK) {//添付ファイル復号化と SD カードへの保存
                Bundle bundle = data.getExtras();
                idvaf = bundle.getInt("key.idvaf");
                idtxtaf = bundle.getString("key.idtxtaf");
                fnamelist = bundle.getStringArrayList("key.fnamelist");
                fpathlist = bundle.getStringArrayList("key.fpathlist");
                fname = bundle.getString("key.fname");
                fpath = bundle.getString("key.fpath");

                // サブディレクトリの作成
                String fullDirName = "";
                fullDirName = getMount_sd() + "/attachdc";
                File dir = new File(fullDirName);
                if (!dir.exists()) {
                    dir.mkdirs();
                }

                // 作業用サブディレクトリの作成
                fullDirName = getMount_sd() + "/dcws";
                dir = new File(fullDirName);
                if (!dir.exists()) {
                    dir.mkdirs();
                }
                fullDirName = getMount_sd() + "/dcws/dc0";
                dir = new File(fullDirName);
                if (!dir.exists()) {
                    dir.mkdirs();
                }
                fullDirName = getMount_sd() + "/dcws/dc1";
                dir = new File(fullDirName);
                if (!dir.exists()) {
                    dir.mkdirs();
                }
                fullDirName = getMount_sd() + "/dcws/dc2";
                dir = new File(fullDirName);
                if (!dir.exists()) {
                    dir.mkdirs();
                }
                fullDirName = getMount_sd() + "/dcws/dc3";
                dir = new File(fullDirName);
                if (!dir.exists()) {
                    dir.mkdirs();
                }
                fullDirName = getMount_sd() + "/dcws/dc4";
```

```
                dir = new File(fullDirName);
                if (!dir.exists()) {
                    dir.mkdirs();
                }
                fullDirName = getMount_sd() + "/dcws/dc5";
                dir = new File(fullDirName);
                if (!dir.exists()) {
                    dir.mkdirs();
                }

                String dcpath0 = getMount_sd() + "/dcws/dc0/";
                String dcpath1 = getMount_sd() + "/dcws/dc1/";
                String dcpath2 = getMount_sd() + "/dcws/dc2/";
                String dcpath3 = getMount_sd() + "/dcws/dc3/";
                String dcpath4 = getMount_sd() + "/dcws/dc4/";
                String dcpath5 = getMount_sd() + "/dcws/dc5/";
                dcpath0 += fname;
                dcpath1 += fname;
                dcpath2 += fname;
                dcpath3 += fname;
                dcpath4 += fname;
                dcpath5 += fname;

//              incfst.close();
                try{

//              copyTransfer(fname, dcpath5);
                File fout = new File(dcpath5);
                fout.getParentFile().mkdir();
                FileOutputStream writer6 = new FileOutputStream(fout);
                in5 = openFileInput(fname);
                while(in5.available()>0){
                        int ib = in5.read(tmpb);
                        writer6.write(tmpb,0,ib);
                }
                in5.close();
                writer6.close();

                if((dcp5.length()>0)&&(dck5.length()>0)){
                        selectfunc( dcp5, dck5, dcpath5, dcpath4);
                }else{
                        copyTransfer(dcpath5,dcpath4);
                }
                if((dcp4.length()>0)&&(dck4.length()>0)){
                        selectfunc( dcp4, dck4, dcpath4, dcpath3);
                }else{
                        copyTransfer( dcpath4, dcpath3);
                }
                if((dcp3.length()>0)&&(dck3.length()>0)){
                        selectfunc( dcp3, dck3, dcpath3, dcpath2);
                }else{
```

```
                    copyTransfer( dcpath3, dcpath2);
                }
                if((dcp2.length()>0)&&(dck2.length()>0)){
                        selectfunc(dcp2, dck2,   dcpath2, dcpath1);
                }else{
                    copyTransfer( dcpath2, dcpath1);
                }
                if((dcp1.length()>0)&&(dck1.length()>0)){
                    selectfunc(dcp1, dck1,   dcpath1, dcpath0);
                }else{
                    copyTransfer( dcpath1, dcpath0);
                }

                fullDirName = getMount_sd() + "/attachdc/" + fname;
                copyTransfer( dcpath0, fullDirName);

                }catch(Exception e) {

                }

                Toast.makeText(this,
              String.format("ファイル%s を、 sdcard-attachdc に保存しました。",
bundle.getString("key.fname")),
              Toast.LENGTH_SHORT).show();

        }

  if (requestCode == 2 && resultCode == RESULT_OK) {//メール返信結果処理
        Bundle bundle = data.getExtras();
            /*
    Toast.makeText(this,
      String.format("こんにちは、%s さん！", bundle.getString("key.name")),
      Toast.LENGTH_SHORT).show();
        */
  }
  }




}

package yu.com.pcs.jp.sumaho.cg5mail;


import yu.com.pcs.jp.sumaho.cg5mail.R;

import java.io.File;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
```

```
import java.util.Map;
import java.util.Random;


import android.app.Activity;
import android.content.ContentValues;
import android.content.Intent;
import android.database.Cursor;
import android.database.sqlite.SQLiteCursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.os.Environment;
import android.view.Menu;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.ExpandableListView;
import android.widget.ExpandableListView.OnChildClickListener;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.ListView;
import android.widget.ExpandableListView.OnGroupClickListener;
import android.widget.SimpleExpandableListAdapter;
import android.widget.TextView;
import android.widget.Toast;


public class MailViewAttachActivity extends Activity {
	//		private AttachDatabaseHelper attachhelper = null;
	int i;

	Integer idvaf = 0;
	String idtxtaf = "0";
	ArrayList<String> fnamelist;// = new ArrayList<String>();
	ArrayList<String> fpathlist;// = new ArrayList<String>();
	String fname;
	String fpath;
	private ListView listview;

	String savestr = "";

	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
	    setContentView(R.layout.activity_mail_view_attach);

		MailViewAttachData attachData =
(MailViewAttachData)getIntent().getSerializableExtra("mailviewattachData");

		idvaf = attachData.getIdv();
		idtxtaf = attachData.getIdtxt();
```

```
                fnamelist = attachData.getFNameList();
                fpathlist = attachData.getFPathList();
                fname = attachData.getFName();
                fpath = attachData.getFPath();

            listview = (ListView)findViewById(R.id.elvat);

            //アダプターの作成
        final ArrayAdapter<String> arrayadapter = new ArrayAdapter<String>
             (this, android.R.layout.simple_list_item_1,fnamelist);
                //アダプターをリストビューにセット
                listview.setAdapter(arrayadapter);

                listview.setOnItemClickListener(
                new AdapterView.OnItemClickListener() {
                public void onItemClick(AdapterView<?> av,
                                View view, int position, long id){
                        savestr = (String)((TextView)view).getText();
//                      arrayadapter.remove((String)((TextView)view).getText());
                }
            }
        );

        Button btn1 = (Button) findViewById(R.id.btn1);
            btn1.setOnClickListener(new OnClickListener() {
                    @Override//もどる
                    public void onClick(View v) {
                            onCancel();
                    }
            });


            Button btn2 = (Button) findViewById(R.id.btn2);
            btn2.setOnClickListener(new OnClickListener() {
                    @Override//保存
                    public void onClick(View v) {
                            onSave();
                    }
            });
        }

  private void onCancel() {//一覧終了
            Intent data = new Intent();
       Bundle bundle = new Bundle();

       bundle.putInt("key.idvaf", idvaf);
       bundle.putString("ket.idtxtaf", idtxtaf);
       bundle.putStringArrayList("key.fnamelist",fnamelist);
       bundle.putStringArrayList("key.fpathlist",fpathlist);
       bundle.putString("key.fname",fname);
       bundle.putString("key.fpath", fpath);
```

```
        data.putExtras(bundle);

        // setResult() で bundle を載せた
        // 送る Intent data をセットする

        // 第一引数は…Activity.RESULT_OK,
        // Activity.RESULT_CANCELED など
        setResult(RESULT_CANCELED, data);

        // finish() で終わらせて
        // Intent data を送る
        finish();
    }

    private void onSave() {//一覧終了
            Intent data = new Intent();
        Bundle bundle = new Bundle();

        bundle.putInt("key.idvaf", idvaf);
        bundle.putString("ket.idtxtaf", idtxtaf);
        bundle.putStringArrayList("key.fnamelist",fnamelist);
        bundle.putStringArrayList("key.fpathlist",fpathlist);
        bundle.putString("key.fname",savestr);
        bundle.putString("key.fpath", fpath);

        data.putExtras(bundle);

        // setResult() で bundle を載せた
        // 送る Intent data をセットする

        // 第一引数は…Activity.RESULT_OK,
        // Activity.RESULT_CANCELED など
        setResult(RESULT_OK, data);

        // finish() で終わらせて
        // Intent data を送る
        finish();
    }


}
package yu.com.pcs.jp.sumaho.cg5mail;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;

public class MailViewAttachData implements Serializable {
        // シリアライズバージョン ID
        private static final long serialVersionUID = 212993500286484661L;
```

```
        // 各データ
          Integer idvaf = 0;
          String idtxtaf = "0";
          ArrayList<String> fnamelist = new ArrayList<String>();
          ArrayList<String> fpathlist = new ArrayList<String>();
          String fname = "fname";
          String fpath = "fpath";

         /**
         * コンストラクタでデータを保存
         */
        public MailViewAttachData(Integer idv , String idtxt, ArrayList<String> fnamelist,
ArrayList<String> fpathlist,
                                String fname, String fpath){
                this.idvaf = idv;
                this.idtxtaf = idtxt;
                this.fnamelist = fnamelist;
                this.fpathlist = fpathlist;
                this.fname = fname;
                this.fpath = fpath;
        }

        /**
         * ゲッター
         * @return
         * @return 保存されているデータ
         */
        public Integer getIdv() {
                return idvaf;
        }
        public String getIdtxt() {
                return idtxtaf;
        }
        public ArrayList<String> getFNameList() {
                return fnamelist;
        }
        public ArrayList<String> getFPathList() {
                return fpathlist;
        }
        public String getFName() {
                return fname;
        }
        public String getFPath() {
                return fpath;
        }

        /**
         * セッター
         */
    public void setIdv(Integer idvaf) {
                this.idvaf = idvaf;
```

```
	}
	public void setIdtxt(String idtxtaf) {
		this.idtxtaf = idtxtaf;
	}
	public void	setFNameList(ArrayList<String> fnamelist) {
		this.fnamelist = fnamelist;
	}
	public void	setFPathList(ArrayList<String> fpathlist) {
		this.fpathlist = fpathlist;
	}
	public void	setFName(String fname) {
		this.fname = fname;
	}
	public	void setFPath(String fpath) {
		this.fpath = fpath;
	}

}

package yu.com.pcs.jp.sumaho.cg5mail;

import yu.com.pcs.jp.sumaho.cg5mail.ListItem;
import yu.com.pcs.jp.sumaho.cg5mail.MyListAdapter;
import yu.com.pcs.jp.sumaho.cg5mail.R;

import java.io.IOException;
import java.util.ArrayList;
import java.util.Enumeration;
import java.util.Properties;
import java.util.Random;

import javax.mail.Address;
import javax.mail.Authenticator;
import javax.mail.BodyPart;
import javax.mail.Folder;
import javax.mail.Header;
import javax.mail.Message;
import javax.mail.MessagingException;
import javax.mail.Multipart;
import javax.mail.PasswordAuthentication;
import javax.mail.Session;
import javax.mail.Store;
import javax.mail.internet.MimeUtility;

import android.app.Activity;
import android.content.ContentValues;
import android.content.Intent;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.os.Environment;
```

```
import android.util.Log;
import android.view.GestureDetector;
import android.view.GestureDetector.SimpleOnGestureListener;
import android.view.Menu;
import android.view.MenuItem;
import android.view.MotionEvent;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ListView;
import android.widget.Toast;

public class MainActivity extends Activity {
   private MailDatabaseHelper mailhelper = null;
   private InitDatabaseHelper initdatahelper = null;
            Integer dc = 0;
            Integer dmc = 2;
            Integer dcount = 0;

            Integer ia, ip;
            String sa, sp;
            ArrayList<Integer> iarray = new ArrayList<Integer>();
            ArrayList<String> sarray = new ArrayList<String>();

            ////////////////////////////////////////////
//          Integer idv ;
            //String idtxt;
            String userid ;
            String address ;
            String password;
            String imaphost;
            String imapport;
            String smtphost;
            String smtpport;
            String pophost;
            String popport;
            String download;
            String memo ;
            Integer totaldl = 0;
   ////////////////////////////////
   private GestureDetector gestureDetector;
   private View.OnTouchListener gestureListener;
   ////////////////////////////////////

   Integer idv = 0;
   String idtxt= "0";
   String attach = "attach";
   String subject = "subject";
   String addressfrom = "addressfrom";
   String addressto = "addressto";
   String date = "date";
   Integer size = 0;
```

```
String priority = "priority";
String read = "read";
String state = "state";
Integer messagenum = 0;
String flag = "flag";
String xmailer = "";
byte[] alldata = null;
Integer i = 0;
Integer j = 0;

@Override
public void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.activity_main);

   mailhelper = new MailDatabaseHelper(this);
   SQLiteDatabase maildb = mailhelper.getWritableDatabase();
   maildb.delete("mailTbl",null,null);

    ////////////////////////////////////
   initdatahelper = new InitDatabaseHelper(this);
   StringBuilder sql = new StringBuilder();
   sql.append(" SELECT");
   sql.append(" id");
   sql.append(" ,idtxt");
   sql.append(" ,userid");
   sql.append(" ,address");
   sql.append(" ,password");
   sql.append(" ,imaphost");
   sql.append(" ,imapport");
   sql.append(" ,smtphost");
   sql.append(" ,smtpport");
   sql.append(" ,pophost");
   sql.append(" ,popport");
   sql.append(" ,download");
   sql.append(" ,memo");
   sql.append(" ,totaldl");
   sql.append(" FROM initTbl;");
   SQLiteDatabase initdb = initdatahelper.getReadableDatabase();
        Cursor cursor = initdb.rawQuery(sql.toString(), null);
   cursor.move(1);                                    //最初の登録 ID を優先する。
   idv =    cursor.getPosition();
   idtxt = cursor.getString(1);
   userid = cursor.getString(2);
   address = cursor.getString(3);
   password = cursor.getString(4);
   imaphost = cursor.getString(5);
   imapport = cursor.getString(6);
   smtphost = cursor.getString(7);
   smtpport = cursor.getString(8);
   pophost = cursor.getString(9);
```

```
        popport = cursor.getString(10);
        download = cursor.getString(11);
        memo     = cursor.getString(12);
        totaldl = cursor.getInt(13);
        initdb.close();
        dmc = Integer.parseInt(download);
        ////////////////////////////////////////////////////
            final String IMAPHOST = imaphost;
            final String USERADDR = address;
            final String PASSWORD = password;
            Properties properties = System.getProperties();
            Session session = Session.getInstance(properties, null);
            Store store = null;

            maildb = mailhelper.getWritableDatabase();
            ContentValues cv = new ContentValues();

////////////////////////////////////////////////////////
ip = -1;
for(i=0; i<sarray.size(); i++){
sa = sarray.get(i);
if(sa.equals(address)){
ip = i;
}
ia = iarray.get(ip);
}
if(ip == -1){
ip = sarray.size();
ia = 0;
sarray.add(address);
iarray.add(ia);
}
////////////////////////////////////////////////////////
            try {
            store = session.getStore("imaps");
            store.connect(IMAPHOST,             USERADDR, PASSWORD);
            //  通常の受信フォルダにアクセスる場合は以下固定
            //Folder folder = store.getFolder("INBOX");
            // IMAP の場合はラベル名を指定すればそのラベルのメールが取得出来る
            // (POP3 の場合はエラーが発生します)
            Folder folder = store.getFolder("INBOX");
            folder.open(Folder.READ_ONLY);
            Integer mc = folder.getMessageCount();

            Message[] messages;
            if(mc < dmc){
                    messages = folder.getMessages(1, mc);
            }else{
                    messages = folder.getMessages(mc-dmc+1, mc);
            }
            dcount += 1;
```

```
        Address[] address;
        String xmailer;
String shn = "";
String shv = "";
        // メッセージ件数分
        //for(int i = 0; i < messages.length; i++) {
        for(j = 0; j<messages.length && j<dmc; j++){
                i = messages.length - j - 1;

                ia++;

                cv.put("attach", "");
            cv.put("subject", messages[i].getSubject());
            address = messages[i].getFrom();
            cv.put("addressfrom", MimeUtility.decodeText(address[0].toString()));
            address = messages[i].getRecipients(Message.RecipientType.TO);
            cv.put("addressto", MimeUtility.decodeText(address[0].toString()));
            cv.put("date", messages[i].getSentDate().toString());
            cv.put("size", messages[i].getSize());
            cv.put("priority", "");
            cv.put("read", "");
            cv.put("state", "");
            cv.put("messagenum", messages[i].getMessageNumber());
            cv.put("flag",    messages[i].getFlags().toString());

            cv.put("xmailer", "");
            Enumeration<Header> headers = messages[i].getAllHeaders();
            while (headers.hasMoreElements()) {
            Header h = headers.nextElement();
            shn = "";
            shv = "";
            shn = h.getName();
            shv = h.getValue();
            if(shn.equals("X-Mailer") && (shv.indexOf("PCS")!= -1)){
                cv.put("xmailer",    shv);
            }
            }

            cv.put("alldata", messages[i].getContent().toString());

            maildb.insert("mailTbl", null, cv);

                }
        folder.close(false);
                }
        catch (Exception e) {
                e.printStackTrace();
                }
        finally {
                try {
```

```
                        if (store != null) {
                                store.close();
                                }
                        }
        catch (MessagingException e) {
                e.printStackTrace();
                }
                }

        iarray.set(ip, ia);
        maildb.close();

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


ListView elvm = (ListView) findViewById(R.id.elvm);
ArrayList<ListItem> data = new ArrayList<ListItem>();

  sql = new StringBuilder();
sql.append(" SELECT");
sql.append(" id");
sql.append(" ,idtxt");
sql.append(" ,attach");
sql.append(" ,subject");
sql.append(" ,addressfrom");
sql.append(" ,addressto");
sql.append(" ,date");
sql.append(" ,size");
sql.append(" ,priority");
sql.append(" ,read");
sql.append(" ,state");
sql.append(" ,messagenum");
sql.append(" ,flag");
sql.append(" ,xmailer");
sql.append(" ,alldata");
sql.append(" FROM mailTbl;");
maildb = mailhelper.getReadableDatabase();

//rawQuery メソッドでデータを取得
try{
        cursor = maildb.rawQuery(sql.toString(), null);
       //TextView に表示
       while (cursor.moveToNext()){
         ListItem item = new ListItem();
         item.setId((new Random()).nextLong());
         item.setSubject(shortSubject(cursor.getString(3)));
         item.setDate(shortDate(cursor.getString(6)));
         item.setFrom(cursor.getString(4));
         data.add(item);
       }
}finally{
```

```
        maildb.close();
            }

MyListAdapter adapter = new MyListAdapter(this, data, R.layout.list_item);
elvm.setAdapter(adapter);

//////////////////////////////////////////////////////
/*
public void onItemClick(AdapterView<?> arg0, View arg1, int arg2, long arg3) {
    // TODO 自動生成されたメソッド・スタブ
    Log.i("TAG", "::" + "リストがタッチされた 2" );
}
*/

class MyGestureDetector extends SimpleOnGestureListener {
    @Override
    public boolean onDoubleTap(MotionEvent event) {
            Log.d("TAG", "ダブルタップが発生した。");
            dc = 1;
            /*return*/ super.onDoubleTap(event);
            return false;
    }


    @Override
    public boolean onDoubleTapEvent(MotionEvent e) {
        Log.v("INFO", "onDoubleTapEvent");
        dc = 1;
        return false;
    }

    @Override
    public boolean onDown(MotionEvent arg0) {
        Log.v("INFO", "onDown");
        dc=1;
        return false;
    }

    @Override
    public boolean onFling(MotionEvent e1, MotionEvent e2, float velocityX, float velocityY) {
        Log.v("INFO", "onFling");
        return false;
    }

    @Override
    public boolean onScroll(MotionEvent e1, MotionEvent e2, float distanceX, float distanceY) {
        Log.v("INFO", "onScroll");
        return false;
    }

    @Override
```

```
        public void onShowPress(MotionEvent e) {
            Log.v("INFO", "onShowPress");
            return;
        }

      //長押し時に呼ばれる
        public void onLongPress(MotionEvent e) {
                Log.v("INFO","LongPress");
                return ;
        }

        @Override
        public boolean onSingleTapUp(MotionEvent e) {
            Log.v("INFO", "onSingleTapUp");
            return false;
        }

        @Override
        public boolean onSingleTapConfirmed(MotionEvent e) {
            Log.v("INFO", "onSingleTapConfirmed");
            return false;
        }
    }

    gestureDetector = new GestureDetector(new MyGestureDetector());
    gestureListener = new View.OnTouchListener() {
        public boolean onTouch(View v, MotionEvent event) {
                return gestureDetector.onTouchEvent(event);
        }
    };

        elvm.setOnTouchListener(gestureListener);

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

    elvm.setOnItemClickListener(
        new AdapterView.OnItemClickListener() {
                public void onItemClick(AdapterView<?> av,
                        View view, int position, long id) {
                        if(dc ==1){
                                StringBuilder sql = new StringBuilder();
                                sql.append(" SELECT");
                                sql.append(" id");
                                sql.append(" ,idtxt");
                                sql.append(" ,attach");
                                sql.append(" ,subject");
                                sql.append(" ,addressfrom");
                                sql.append(" ,addressto");
                                sql.append(" ,date");
                                sql.append(" ,size");
                                sql.append(" ,priority");
```

```
                                    sql.append(" ,read");
                                    sql.append(" ,state");
                                    sql.append(" ,messagenum");
                                    sql.append(" ,flag");
                                    sql.append(" ,xmailer");
                                    sql.append(" ,alldata");
                                    sql.append(" FROM mailTbl;");
                                    SQLiteDatabase maildb = mailhelper.getReadableDatabase();
                                    Cursor cursor = maildb.rawQuery(sql.toString(), null);
                                    cursor.move(position+1);
                                    idv =    cursor.getPosition();
                                    idtxt = cursor.getString(1);;
                                    attach = cursor.getString(2);;
                                    subject = cursor.getString(3);
                                    addressfrom = cursor.getString(4);
                                    addressto = cursor.getString(5);
                                    date = cursor.getString(6);
                                    size = cursor.getInt(7);
                                    priority = cursor.getString(8);
                                    read = cursor.getString(9);
                                    state = cursor.getString(10);
                                    messagenum = cursor.getInt(11);
                                    flag = cursor.getString(12);
                                    xmailer = cursor.getString(13);
                                    alldata = cursor.getBlob(14);

                                    mailview();

                            }
                            dc = 0;
                            return ;
                    }
            }
      );//end ofelvm.setOnItemClickListener(
 }//end of onCreate

public String shortSubject(String dt){
        String ssbj;

        int l = dt.length();
        if(l>13){
                ssbj = dt.substring(0,13);
         }
        else{
                ssbj = dt;
        }

         return ssbj;

}
```

```
public String shortDate(String dt){
        return dt;
        /*
                String sdt = "";
                int dl = dt.length();
                int cp = 0;
                String num = "";
                switch(dl){
                        case 25:
                                sdt = dt.substring(6, 10);
                                String mm = dt.substring(2,5);
                                if(mm == "Jan"){num = "01";}
                                    else {if(mm == "Feb"){num = "02";}
                                    else {if(mm == "Mar"){num = "03";}
                                    else {if(mm == "Apr"){num = "04";}
                                    else {if(mm == "May"){num = "05";}
                                    else {if(mm == "Jun"){num = "06";}
                                    else {if(mm == "Jul"){num = "07";}
                                    else {if(mm == "Aug"){num = "08";}
                                    else {if(mm == "Sep"){num = "09";}
                                    else {if(mm == "Oct"){num = "10";}
                                    else {if(mm == "Nov"){num = "11";}
                                    else {if(mm == "Dec"){num = "12";}
                                }}}}}}}}}}}}
                                sdt += "/" + num + "/0"+ dt.substring(0,1);
                                cp = dt.indexOf(":");
                                sdt += " " + dt.substring(cp-2,cp-2+8);
                                break;
                        case 26:
                                sdt = dt.substring(7, 11);
                                mm = dt.substring(3,6);
                                if(mm == "Jan"){num = "01";}
                                    else {if(mm == "Feb"){num = "02";}
                                    else {if(mm == "Mar"){num = "03";}
                                    else {if(mm == "Apr"){num = "04";}
                                    else {if(mm == "May"){num = "05";}
                                    else {if(mm == "Jun"){num = "06";}
                                    else {if(mm == "Jul"){num = "07";}
                                    else {if(mm == "Aug"){num = "08";}
                                    else {if(mm == "Sep"){num = "09";}
                                    else {if(mm == "Oct"){num = "10";}
                                    else {if(mm == "Nov"){num = "11";}
                                    else {if(mm == "Dec"){num = "12";}
                                }}}}}}}}}}}}
                                sdt += "/" + num + "/"+ dt.substring(0,2);
                                cp = dt.indexOf(":");
                                sdt += " " + dt.substring(cp-2,cp-2+8);
                                break;

                        case 28:
```

```
            sdt = dt.substring(24, 28);
            mm = dt.substring(4,7);
            if(mm.equals("Jan")){num = "01";}
            else {if(mm.equals("Feb")){num = "02";}
            else {if(mm.equals("Mar")){num = "03";}
            else {if(mm.equals("Apr")){num = "04";}
            else {if(mm.equals("May")){num = "05";}
            else {if(mm.equals("Jun")){num = "06";}
            else {if(mm.equals("Jul")){num = "07";}
            else {if(mm.equals("Aug")){num = "08";}
            else {if(mm.equals("Sep")){num = "09";}
            else {if(mm.equals("Oct")){num = "10";}
            else {if(mm.equals("Nov")){num = "11";}
            else {if(mm.equals("Dec")){num = "12";}
            }}}}}}}}}}}}
            sdt += "/" + num + "/"+ dt.substring(8,10);
            cp = dt.indexOf(":");
            sdt += " " + dt.substring(cp-2,cp-2+8);
            break;

        case 29:
            sdt = dt.substring(12, 16);
            mm = dt.substring(8,11);
            if(mm == "Jan"){num = "01";}
                else {if(mm == "Feb"){num = "02";}
                else {if(mm == "Mar"){num = "03";}
                else {if(mm == "Apr"){num = "04";}
                else {if(mm == "May"){num = "05";}
                else {if(mm == "Jun"){num = "06";}
                else {if(mm == "Jul"){num = "07";}
                else {if(mm == "Aug"){num = "08";}
                else {if(mm == "Sep"){num = "09";}
                else {if(mm == "Oct"){num = "10";}
                else {if(mm == "Nov"){num = "11";}
                else {if(mm == "Dec"){num = "12";}
            }}}}}}}}}}}}
            sdt += "/" + num + "/"+ dt.substring(5,7);
            cp = dt.indexOf(":");
            sdt += " " + dt.substring(cp-2,cp-2+8);
            break;
        case 30:
            sdt = dt.substring(11, 15);
            mm = dt.substring(7,10);
            if(mm == "Jan"){num = "01";}
                        else {if(mm == "Feb"){num = "02";}
                        else {if(mm == "Mar"){num = "03";}
                        else {if(mm == "Apr"){num = "04";}
                        else {if(mm == "May"){num = "05";}
                        else {if(mm == "Jun"){num = "06";}
                        else {if(mm == "Jul"){num = "07";}
                        else {if(mm == "Aug"){num = "08";}
```

```
                    else {if(mm == "Sep"){num = "09";}
                    else {if(mm == "Oct"){num = "10";}
                    else {if(mm == "Nov"){num = "11";}
                    else {if(mm == "Dec"){num = "12";}
                }}}}}}}}}}
                sdt += "/" + num + "/0"+ dt.substring( 5,6);
                cp = dt.indexOf(":");
                sdt += " " + dt.substring(cp-2,cp-2+8);
                break;
        case 31:
                sdt = dt.substring(12, 16);
                mm = dt.substring(8,11);
                if(mm == "Jan"){num = "01";}
                    else {if(mm == "Feb"){num = "02";}
                    else {if(mm == "Mar"){num = "03";}
                    else {if(mm == "Apr"){num = "04";}
                    else {if(mm == "May"){num = "05";}
                    else {if(mm == "Jun"){num = "06";}
                    else {if(mm == "Jul"){num = "07";}
                    else {if(mm == "Aug"){num = "08";}
                    else {if(mm == "Sep"){num = "09";}
                    else {if(mm == "Oct"){num = "10";}
                    else {if(mm == "Nov"){num = "11";}
                    else {if(mm == "Dec"){num = "12";}
                }}}}}}}}}}
                sdt += "/" + num + "/"+ dt.substring( 5,7);
                cp = dt.indexOf(":");
                sdt += " " + dt.substring(cp-2,cp-2+8);
                break;
        case 32:
        case 34:
        case 35:
                sdt = dt.substring(12, 16);
                mm = dt.substring(8,11);
                if(mm == "Jan"){num = "01";}
                    else {if(mm == "Feb"){num = "02";}
                    else {if(mm == "Mar"){num = "03";}
                    else {if(mm == "Apr"){num = "04";}
                    else {if(mm == "May"){num = "05";}
                    else {if(mm == "Jun"){num = "06";}
                    else {if(mm == "Jul"){num = "07";}
                    else {if(mm == "Aug"){num = "08";}
                    else {if(mm == "Sep"){num = "09";}
                    else {if(mm == "Oct"){num = "10";}
                    else {if(mm == "Nov"){num = "11";}
                    else {if(mm == "Dec"){num = "12";}
                }}}}}}}}}}
                sdt += "/" + num + "/"+ dt.substring( 5,7);
                cp = dt.indexOf(":");
                sdt += " " + dt.substring(cp-2,cp-2+8);
                break;
```

```
		case 36:
			sdt = dt.substring(11, 15);
			mm = dt.substring(7,10);
			if(mm == "Jan"){num = "01";}
				else {if(mm == "Feb"){num = "02";}
				else {if(mm == "Mar"){num = "03";}
				else {if(mm == "Apr"){num = "04";}
				else {if(mm == "May"){num = "05";}
				else {if(mm == "Jun"){num = "06";}
				else {if(mm == "Jul"){num = "07";}
				else {if(mm == "Aug"){num = "08";}
				else {if(mm == "Sep"){num = "09";}
				else {if(mm == "Oct"){num = "10";}
				else {if(mm == "Nov"){num = "11";}
				else {if(mm == "Dec"){num = "12";}
			}}}}}}}}}}}}
			sdt += "/" + num + "/0"+ dt.substring( 5,6);
			cp = dt.indexOf(":");
			sdt += " " + dt.substring(cp-2,cp-2+8);
			break;
		case 37:
		case 43:
			sdt = dt.substring(12, 16);
			mm = dt.substring(8,11);
			if(mm == "Jan"){num = "01";}
				else {if(mm == "Feb"){num = "02";}
				else {if(mm == "Mar"){num = "03";}
				else {if(mm == "Apr"){num = "04";}
				else {if(mm == "May"){num = "05";}
				else {if(mm == "Jun"){num = "06";}
				else {if(mm == "Jul"){num = "07";}
				else {if(mm == "Aug"){num = "08";}
				else {if(mm == "Sep"){num = "09";}
				else {if(mm == "Oct"){num = "10";}
				else {if(mm == "Nov"){num = "11";}
				else {if(mm == "Dec"){num = "12";}
			}}}}}}}}}}}}
			mm = dt.substring(5,6);
			if(mm != " "){
				sdt += "/" + num + "/"+ dt.substring( 5,7);
			}
			else{
				sdt += "/" + num + "/0"+ dt.substring( 6,7);
			}
			cp = dt.indexOf(":");
			sdt += " " + dt.substring(cp-2,cp-2+8);
			break;
default:
	sdt = dt;

 }
```

```
                return sdt;
                */
        }

        public void mailview() {
                MailData mailData = new MailData(idv , idtxt, attach, subject, addressfrom,
                                addressto, date, size , priority , read, state, messagenum, flag,
xmailer, alldata);

                Intent i = new Intent(this, yu.com.pcs.jp.sumaho.cg5mail.MailViewActivity.class);
                i.putExtra("mailData", mailData);
                this.startActivityForResult(i, 1);
        }


  @Override
  public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.option_menu, menu);
    return true;
  }

  @Override
  public boolean onOptionsItemSelected(MenuItem item) {
    Toast toast = Toast.makeText(this, item.getTitle(), Toast.LENGTH_LONG);
    toast.show();

    switch(item.getItemId()){
    /*
    case R.id.item1:
        //search

        break;
    case R.id.item2:
        //refresh

        break;
    case R.id.item3:
        //sort

        break;
        */
    case R.id.item4:
        DummyEdit(null);
        break;

    case R.id.item5:
        //address book
            Intent i = new Intent(this, yu.com.pcs.jp.sumaho.cg5mail.AddrListShowActivity.class);
            startActivity(i);
        break;
```

```
        case R.id.item6:
            //tools
            tools(null);
        //  i = new Intent(this, yu.com.pcs.jp.sumaho.cg3mail.InitListShowActivity.class);
            //      startActivity(i);
                    break;
        }

        return true;
    }

    public void onEnd(View view) {
            finish();
        }

    public void onRest(View view) {
                Toast.makeText(this, "休止します。",
                  Toast.LENGTH_SHORT).show();

                moveTaskToBack(true);
                }

public void onMailDL(View view) {
            InitData initData = new InitData(idv , idtxt, userid, address,
                                password, imaphost,imapport, smtphost , smtpport , pophost, popport,
download, memo, totaldl);
            Intent i = new Intent(this, yu.com.pcs.jp.sumaho.cg5mail.InitListDLSelectActivity.class);
            i.putExtra("initData", initData);
            this.startActivityForResult(i, 2);
            }

private static String getText(Object content)
            throws IOException, MessagingException {
            String text = null;
            StringBuffer sb = new StringBuffer();
            if (content instanceof String) {
                        sb.append((String) content);
            }
            else if (content instanceof Multipart) {
                                Multipart mp = (Multipart) content;
                                for (int i = 0; i < mp.getCount(); i++) {
                                        BodyPart bp = mp.getBodyPart(i);
                                        sb.append(getText(bp.getContent()));
                                }
                        }
            text = sb.toString();
            return text;
}
```

```
                    /*
public boolean onSort(View view) {

            // Checks if external storage is available for read and write
            //public boolean isExternalStorageWritable() {
        String state = Environment.getExternalStorageState();
        if (Environment.MEDIA_MOUNTED.equals(state)) {
            return true;
        }
          //          return false;
           // }

            // Checks if external storage is available to at least read
            // public boolean isExternalStorageReadable() {
          //          String sstate = Environment.getExternalStorageState();
        if (Environment.MEDIA_MOUNTED.equals(state) ||
            Environment.MEDIA_MOUNTED_READ_ONLY.equals(state)) {
            return true;
        }
        return false;
   //}

}
*/

public void onWriteNew(View view) {
        addressfrom = "";

        MailData mailData2 = new MailData(idv , idtxt, attach, subject, addressfrom,
                        addressto, date, size , priority , read, state, messagenum, flag, xmailer,
alldata);

        Intent i = new Intent(this, yu.com.pcs.jp.sumaho.cg5mail.MailRetEditActivity.class);
        i.putExtra("mailData2", mailData2);
        this.startActivityForResult(i, 1);
}

public void DummyEdit(View view) {
        Intent i = new Intent(this, yu.com.pcs.jp.sumaho.cg5mail.DummyEditActivity.class);
    startActivity(i);
}

public void tools(View view) {
        Intent i = new Intent(this, yu.com.pcs.jp.sumaho.cg5mail.InitListShowActivity.class);
    startActivity(i);
}

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
```

```
        ////////////////////////////////////////////////////////////////////////////
        if (requestCode == 1 && resultCode == RESULT_OK) {//メール作成の結果処理
                Bundle bundle = data.getExtras();
        }

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

        if (requestCode == 2 && resultCode == RESULT_OK) {//メール DL 処理
                Bundle bundle = data.getExtras();
                final String sdmc = bundle.getString("key.download");
        dmc = Integer.valueOf(sdmc);
        Store store = null;
        ContentValues cv = new ContentValues();
        SQLiteDatabase maildb = mailhelper.getWritableDatabase();
        final String PASSWORD = bundle.getString("key.password");
        final String USER = bundle.getString("key.address");
        final String HOST = bundle.getString("key.imaphost");
        final String USERID = bundle.getString("key.userid");
                final String spophost = bundle.getString("key.pophost");

////////////////////////////////////////////////////////
ip = -1;
for(i=0; i<sarray.size(); i++){
sa = sarray.get(i);
if(sa.equals(USER)){
ip = i;
ia = iarray.get(ip);
}
}
if(ip == -1){
ip = sarray.size();
ia = 0;
sarray.add(USER);
iarray.add(ia);
}
////////////////////////////////////////////////////////

        if(!HOST.equals("")){//use imaps
                Properties properties = System.getProperties();
                Session session = Session.getInstance(properties, null);
                try {
                store = session.getStore("imaps");
                store.connect(HOST,        USER, PASSWORD);
                //  通常の受信フォルダにアクセスる場合は以下固定
                //Folder folder = store.getFolder("INBOX");
                // IMAP の場合はラベル名を指定すればそのラベルのメールが取得出来る
                // (POP3 の場合はエラーが発生します)
                Folder folder = store.getFolder("INBOX");
                folder.open(Folder.READ_ONLY);
                Integer mc = folder.getMessageCount();
                Message[] messages;
```

```
        if(mc <= ia){return;}
        mc = mc-ia;
        if(mc <= dmc){
                messages = folder.getMessages(1, mc);
        }else{
                messages = folder.getMessages(mc-dmc+1, mc);
        }
        dcount += 1;
        Address[] address;
        String xmailer;
String shn = "";
String shv = "";
        // メッセージ件数分
for(j = 0; j<messages.length && j<dmc; j++){
                i = messages.length - 1 - j;

                ia++;

                cv.put("attach", "");
            cv.put("subject", messages[i].getSubject());
            address = messages[i].getFrom();
            cv.put("addressfrom", MimeUtility.decodeText(address[0].toString()));
            address = messages[i].getRecipients(Message.RecipientType.TO);
            cv.put("addressto", MimeUtility.decodeText(address[0].toString()));
            cv.put("date", messages[i].getSentDate().toString());
            cv.put("size", messages[i].getSize());
            cv.put("priority", "");
            cv.put("read", "");
            cv.put("state", "");
            cv.put("messagenum", messages[i].getMessageNumber());
            cv.put("flag",    messages[i].getFlags().toString());

            cv.put("xmailer", "");
            Enumeration<Header> headers = messages[i].getAllHeaders();
            while (headers.hasMoreElements()) {
            Header h = headers.nextElement();
            shn = "";
            shv = "";
            shn = h.getName();
            shv = h.getValue();
            if(shn.equals("X-Mailer") && (shv.indexOf("PCS")!= -1)){
                cv.put("xmailer",    shv);
            }
}
cv.put("alldata", messages[i].getContent().toString());
maildb.insert("mailTbl", null, cv);
        }
        folder.close(false);
        iarray.set(ip, ia);

        }
```

```
                catch (Exception e) {
                        e.printStackTrace();
                        }
                finally {
                        try {
                                if (store != null) {
                                        store.close();
                                        }
                                }
                catch (MessagingException e) {
                        e.printStackTrace();
                        }
                        }
        }

        if( HOST.equals("")){//use pop3
                        Properties prop = new Properties();
                    prop.put("mail.host",spophost);
                    prop.put("mail.store.protocol","pop3"); //"pop3"固定
                        try {
                                Session session=Session.getDefaultInstance(prop,new
Authenticator(){//メールサーバとの間に Session を作成
                                        protected PasswordAuthentication
getPasswordAuthentication(){
                                        return new
PasswordAuthentication(USERID,PASSWORD); //適当なユーザ名とパスワードに書換える
                                        }
                                    });
                                store = session.getStore("pop3");//"pop3"固定
                                store.connect(spophost, null, null);

                                Folder inbox = store.getFolder("INBOX");
                                if(inbox==null){
                                        String smes = "NU INBOX";
                                        return;
                                }
                                inbox.open(Folder.READ_ONLY);

                                Message[] messages = inbox.getMessages();
                                Integer mc = inbox.getMessageCount();
                                if(mc <= ia){return;}
                                mc = mc-ia;
                                if(mc <= dmc){
                                        messages = inbox.getMessages(1, mc);
                                }else{
                                        messages = inbox.getMessages(mc-dmc+1, mc);
                                }

                                dcount += 1;
                                Address[] address;
                                String xmailer;
```

```
                        String shn = "";
                        String shv = "";
                        // メッセージ件数分
                        for(j = 0; j<messages.length && j<dmc; j++){
                                i = messages.length - 1 - j;

                                ia++;

                                cv.put("attach", "");
                                cv.put("subject", messages[i].getSubject());
                                address = messages[i].getFrom();
                                cv.put("addressfrom",
MimeUtility.decodeText(address[0].toString()));
                                address                                                 =
messages[i].getRecipients(Message.RecipientType.TO);
                                cv.put("addressto",
MimeUtility.decodeText(address[0].toString()));
                                cv.put("date", messages[i].getSentDate().toString());
                                cv.put("size", messages[i].getSize());
                                cv.put("priority", "");
                                cv.put("read", "");
                                cv.put("state", "");
                                cv.put("messagenum", messages[i].getMessageNumber());
                                cv.put("flag",    messages[i].getFlags().toString());

                                cv.put("xmailer", "");
                                Enumeration<Header>                 headers                 =
messages[i].getAllHeaders();
                                while (headers.hasMoreElements()) {
                                        Header h = headers.nextElement();
                                        shn = "";
                                        shv = "";
                                        shn = h.getName();
                                        shv = h.getValue();
                                        if(shn.equals("X-Mailer")                               &&
(shv.indexOf("PCS")!= -1)){
                                                cv.put("xmailer",    shv);
                                        }
                                }
                                cv.put("alldata", messages[i].getContent().toString());
                                maildb.insert("mailTbl", null, cv);
                        }
                        inbox.close(false);
                        iarray.set(ip, ia);
                }
                catch (Exception e) {
                        e.printStackTrace();
                }
                finally {
                        try {
                                if (store != null) {
```

```
                                        store.close();
                                }
                        }
                        catch (MessagingException e) {
                                e.printStackTrace();
                        }
                }
}

        maildb.close();
        ListView elvm = (ListView) findViewById(R.id.elvm);
    ArrayList<ListItem> ldata = new ArrayList<ListItem>();
    StringBuilder sql = new StringBuilder();
    sql.append(" SELECT");
    sql.append(" id");
    sql.append(" ,idtxt");
    sql.append(" ,attach");
    sql.append(" ,subject");
    sql.append(" ,addressfrom");
    sql.append(" ,addressto");
    sql.append(" ,date");
    sql.append(" ,size");
    sql.append(" ,priority");
    sql.append(" ,read");
    sql.append(" ,state");
    sql.append(" ,messagenum");
    sql.append(" ,flag");
    sql.append(" ,xmailer");
    sql.append(" ,alldata");
    sql.append(" FROM mailTbl;");
    maildb = mailhelper.getReadableDatabase();
    //rawQuery メソッドでデータを取得
    try{
        Cursor cursor = maildb.rawQuery(sql.toString(), null);
        //TextView に表示
        while (cursor.moveToNext()){
          ListItem item = new ListItem();
          item.setId((new Random()).nextLong());
          item.setSubject(shortSubject(cursor.getString(3)));
          item.setDate(shortDate(cursor.getString(6)));
          item.setFrom(cursor.getString(4));
          ldata.add(item);
        }
    }finally{
          maildb.close();
    }
    //作ったデータを、MyListAdapter に渡して表示してもらう。
    MyListAdapter adapter = new MyListAdapter(this, ldata, R.layout.list_item);
    elvm.setAdapter(adapter);
}
```

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


        }
}

package yu.com.pcs.jp.sumaho.cg5mail;

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

import android.app.Activity;
import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.os.Bundle;

import java.io.BufferedInputStream;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;

import yu.com.pcs.jp.sumaho.cg5mail.MailRetEditActivity.MARScipherInstance;
import yu.com.pcs.jp.sumaho.cg5mail.MailRetEditActivity.MARSkeyInstance;
import yu.com.pcs.jp.sumaho.cg5mail.MailViewActivity.cipherInstance;
import yu.com.pcs.jp.sumaho.cg5mail.MailViewActivity.keyInstance;

public class MARSActivity extends Activity{
        String prgn;
        String keyfn;
        String inputfn;
        String outputfn;

        @Override
        protected void onCreate(Bundle savedInstanceState) {
           super.onCreate(savedInstanceState);
                prgn = "";
            keyfn = "";
            inputfn = "";
            outputfn = "";
        }

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

```
//////////////////////MARS DC  ////////////////////////////////////////////

/*  aes.h  */

/*  AES Cipher header file for ANSI C Submissions
*     Lawrence E. Bassham III
*     Computer Security Division
*     National Institute of Standards and Technology
*
*     April 15, 1998
*
*     Modified for IBM submission
*     David Safford
*     4/16/1998
*/

//#include <stdio.h>

/*****************************************************************************
*
*     NIST High Level C API, with some IBM additions
*
*****************************************************************************/

//#define TRUE                1
//#define FALSE               0

//#define DIR_ENCRYPT       0     /*  Are we encrypting?                     */
//#define DIR_DECRYPT       1     /*  Are we decrypting?                     */
//#define MODE_ECB          1     /*  Are we ciphering in ECB mode?          */
//#define MODE_CBC          2     /*  Are we ciphering in CBC mode?          */
//#define MODE_CFB1         3     /*  Are we ciphering in 1-bit CFB mode?    */

//#define BAD_KEY_DIR        -1   /*  Key direction is invalid               */
//#define BAD_KEY_MAT        -2   /*  Key material not of correct length     */
//#define BAD_KEY_INSTANCE -3   /*  Key passed is not valid                */
//#define BAD_CIPHER_MODE   -4   /*  Params struct passed to cipherInit invalid */
//#define BAD_CIPHER_STATE -5   /*  Cipher in wrong state                  */

//typedef unsigned char BYTE;

/* IBM Addition - a DWORD must be 32 bits for this implementation */
//typedef unsigned long DWORD;

/* IBM specific defines: these parameters can be changed                  */
//#define NUM_MIX 8               /* number of mixing rounds per stage     */
//#define NUM_ROUNDS 16           /* number of full core rounds            */
//#define NUM_SETUP 7             /* number of key setup mixing rounds     */

/* IBM specific defines: these parameters are fixed for this implementation */
//#define W              32       /* number of bits in a word              */
```

```
//#define NUM_DATA      4           /* data block size in words                    */
//#define EKEY_DWORDS (2*(NUM_DATA+NUM_ROUNDS))      /* number of subkey words     */

/* IBM modified values                                                           */
//#define MAX_KEY_SIZE (EKEY_DWORDS*8)   /* max ASCII char's needed for a key     */
//#define MAX_IV_SIZE (NUM_DATA*4)        /* max bytes's needed for an IV          */


/*   The structure for key information */
class MARSkeyInstance{
        public int    direction;        /*   Key used for encrypting or decrypting?        */
        public int     keyLen;          /*   Length of the key in BITS                      */
        public int[] keyMaterial = new int[2*(4+16)*8+1];   /*   Raw key data in ASCII      */
        public int[] E = new int[2*(4+16)];     /* IBM addition for mars expanded key        */
}

/*   The structure for cipher information */
class MARScipherInstance {
        public   int   mode;                    /*     MODE_ECB,   MODE_CBC,   or   MODE_CFB1
*/
        public int[] IV = new int[4*4];   /*   initial binary IV BYTE for chaining         */
        public byte[] IVb = new byte[4*4*4];
        public int[] CIV = new int[4];      /*   IBM addition: current IV in binary WORDs      */
        public byte[] CIVb = new byte[32];
}

int s;
MARSkeyInstance keyI;
MARScipherInstance cipherI;
int[] t = new int[4];
int   b, n, i;
int[] x = new int[128/32];
byte bit, bit0, ctBit, carry;

int SWAP_BYTES = 0;

int ob=0, oip=0, obmode =0;
int[] ox = new int[128/32];
int[] pbuf = new int[2048/4];
int[] cbuf = new int[2048/4];
byte[] cbufb = new byte[2048];
byte[] pbufb = new byte[2048];



/*   NIST High level function protoypes   */
//int makeKey(keyInstance *key, BYTE direction, int keyLen, char *keyMaterial);

//int cipherInit(cipherInstance *cipher, BYTE mode, char *IV);

//int blockEncrypt(cipherInstance *cipher, keyInstance *key, BYTE *input,
```

```
//int inputLen, BYTE *outBuffer);

//int blockDecrypt(cipherInstance *cipher, keyInstance *key, BYTE *input,
//int inputLen, BYTE *outBuffer);

/*****************************************************************************
*
*       IBM Low Level (WORD Oriented) API
*
*****************************************************************************/

/* setup a mars expanded key
*
* k    (input) is the number of words in the key
* kp (input) is a pointer to the array of k key words
* ep (output) is a pointer to an array of EKEY_WORDS expanded subkey WORDs
*/
//int mars_setup(int k, DWORD *kp, DWORD *ep);

/* The basic mars encryption of one block (of NUM_DATA DWORDS) */
//void mars_encrypt(DWORD *in, DWORD *out, DWORD *ep);

/* mars decryption is simply encryption in reverse */
//void mars_decrypt(DWORD *in, DWORD *out, DWORD *ep);




/* mars.c optimized C code - copyright(c) 1998 IBM
*
* This code implements both the NIST high level C API (version 5)
* and an underlying IBM defined low level (DWORD oriented) C API.
*/

/* Revisions log:
*
*     Aug 1999, Shai - the "tweaked" key schedule
*     Apr 1998, Dave && Shai - new key scheduling, new sbox, NIST API
*     Mar 1998, Shai Halevi    - adapted to the latest variant of mars
*     Feb 1998, Dave Safford - created
*/

/* Compilation using pcg's version of gcc:
*       gcc -Wall -pedantic -c -O6 -fomit-frame-pointer -mcpu=pentiumpro
*             -DINTEL_GCC tests.c mars-opt.c
*
* Compilation using xlc on AIX:
*       xlc -c -O3 -DAIX_XLC mars-opt.c
*
* Compilation using Borland C++ 5.0 from a DOS command line:
*       bcc32 -Oi -6 -v -A -a4 -O2 -DKAT tests.c mars-opt.c
```

```
*
* Useful compilation defines:
*       NO_MIX         - do not execute the mixing phases (core only)
*       INTEL_GCC      - optimized for pentiumpro and djgpp/gcc compiler
*       AIX_XLC        - optimized for powerpc/AIX with xlc compiler
*       SWAP_BYTES - force endian conversion (eg __BYTE_ORDER not supported)
*       IVT            - add intermediate values test outputs (this slows
*                        down encryption and decryption significantly)
*/
//#include "stdafx.h"
//#include "aes.h"

/* The low level mars routines are completely DWORD oriented, and
* endian neutral. The high level NIST routines provide BYTE oriented
* inputs and outputs, thus raising the endian issue when converting
* between BYTEs and DWORDs. For these conversions, mars assumes
* little endian order. On a big endian machine, we define BSWAP()
* to do the conversions. BSWAP() is about the best you can do in C.
* Real implementations will undoubtedly use inline ASM, as most risc
* machines can do this in one instruction.
*
* Make a best guess on platform endianness; This works on linux, AIX,
* and W95. For other platforms, you may have to manually define SWAP_BYTES
* on a big endian machine if this guessing doesn't work.
*/
/*
#ifdef _AIX
#define SWAP_BYTES
#else
#ifdef __linux__
#include <endian.h>
#ifdef __BYTE_ORDER
#if __BYTE_ORDER == __BIG_ENDIAN
#define SWAP_BYTES
#endif
#endif
#endif
#endif
#ifdef SWAP_BYTES
#define BSWAP(x) ¥
( (( x                ) << 24) | ¥
(((x)&0xff00    ) << 8 ) | ¥
(((x)&0xff0000) >> 8 ) | ¥
(( x                ) >> 24)    )
#else
#define BSWAP(x) (x)
#endif
*/

int BSWAP(int x){
        if(SWAP_BYTES == 1){
```

```
                return ( (( x ) << 24) | (((x)&0xff00   ) << 8 ) | (((x)&0xff0000) >> 8 ) | (( x ) >>
24)   )   ;
        }else{
                return x;
        }
}


/* Some compiler optimizers will recognize certain rotation idioms,
* and reduce them to native rotation instructions. These idioms
* have been found to work for GCC on Intel, and XLC on RS6000.
* Nothing seems to work on Borland C++ 5.0, although we can leave
* out the mask on 'b'.
*/
/*
#ifdef INTEL_GCC
#define LROTATE(a,b) (((a)<<(int)(b)) | ((a)>>(W - (int)(b))))
#define RROTATE(a,b) (((a)>>(int)(b)) | ((a)<<(W - (int)(b))))
#else
#ifdef AIX_XLC
#define LROTATE(a,b) ( ((a)>>(W - (b))) | ((a)<<(b)) )
#define RROTATE(a,b) ( ((a)<<(W - (b))) | ((a)>>(b)) )
#else
#ifdef __BORLANDC__
#define LROTATE(a,b) ( ((a)>>(W - (b))) | ((a)<<(b)) )
#define RROTATE(a,b) ( ((a)<<(W - (b))) | ((a)>>(b)) )
#else
#define LROTATE(a,b) (((a)<<(int)(b&31)) | ((a)>>(W - (int)(b&31))))
#define RROTATE(a,b) (((a)>>(int)(b&31)) | ((a)<<(W - (int)(b&31))))
#endif
#endif
#endif
*/
////////////////////////////////////////
//#define BLOCK_SIZE 128


/* two 8 x 32 sboxes - stored together to save a pointer
* these sboxes were generated with buf[3] = 0x02917d59, as
* chosen by sbox.c, which had the following output:
*     After 54817140, (38023 fail) new min j = 43089241 (0x2917d59)
*     test 0 eval 0.044922 (single bit correlation)
*     test 1 eval 0.033203 (single bit bias)
*     test 2 eval 0.031250 (consecutive bit bias)
*     test 3 eval 0.007813 (parity bias)
*     test 4 eval 0.148438 (avalanche)
*/

static int[] S = {
0x09d0c479, 0x28c8ffe0, 0x84aa6c39, 0x9dad7287,
0x7dff9be3, 0xd4268361, 0xc96da1d4, 0x7974cc93,
```

0x85d0582e, 0x2a4b5705, 0x1ca16a62, 0xc3bd279d,
0x0f1f25e5, 0x5160372f, 0xc695c1fb, 0x4d7ff1e4,
0xae5f6bf4, 0x0d72ee46, 0xff23de8a, 0xb1cf8e83,
0xf14902e2, 0x3e981e42, 0x8bf53eb6, 0x7f4bf8ac,
0x83631f83, 0x25970205, 0x76afe784, 0x3a7931d4,
0x4f846450, 0x5c64c3f6, 0x210a5f18, 0xc6986a26,
0x28f4e826, 0x3a60a81c, 0xd340a664, 0x7ea820c4,
0x526687c5, 0x7eddd12b, 0x32a11d1d, 0x9c9ef086,
0x80f6e831, 0xab6f04ad, 0x56fb9b53, 0x8b2e095c,
0xb68556ae, 0xd2250b0d, 0x294a7721, 0xe21fb253,
0xae136749, 0xe82aae86, 0x93365104, 0x99404a66,
0x78a784dc, 0xb69ba84b, 0x04046793, 0x23db5c1e,
0x46cae1d6, 0x2fe28134, 0x5a223942, 0x1863cd5b,
0xc190c6e3, 0x07dfb846, 0x6eb88816, 0x2d0dcc4a,
0xa4ccae59, 0x3798670d, 0xcbfa9493, 0x4f481d45,
0xeafc8ca8, 0xdb1129d6, 0xb0449e20, 0x0f5407fb,
0x6167d9a8, 0xd1f45763, 0x4daa96c3, 0x3bec5958,
0xababa014, 0xb6ccd201, 0x38d6279f, 0x02682215,
0x8f376cd5, 0x092c237e, 0xbfc56593, 0x32889d2c,
0x854b3e95, 0x05bb9b43, 0x7dcd5dcd, 0xa02e926c,
0xfae527e5, 0x36a1c330, 0x3412e1ae, 0xf257f462,
0x3c4f1d71, 0x30a2e809, 0x68e5f551, 0x9c61ba44,
0x5ded0ab8, 0x75ce09c8, 0x9654f93e, 0x698c0cca,
0x243cb3e4, 0x2b062b97, 0x0f3b8d9e, 0x00e050df,
0xfc5d6166, 0xe35f9288, 0xc079550d, 0x0591aee8,
0x8e531e74, 0x75fe3578, 0x2f6d829a, 0xf60b21ae,
0x95e8eb8d, 0x6699486b, 0x901d7d9b, 0xfd6d6e31,
0x1090acef, 0xe0670dd8, 0xdab2e692, 0xcd6d4365,
0xe5393514, 0x3af345f0, 0x6241fc4d, 0x460da3a3,
0x7bcf3729, 0x8bf1d1e0, 0x14aac070, 0x1587ed55,
0x3afd7d3e, 0xd2f29e01, 0x29a9d1f6, 0xefb10c53,
0xcf3b870f, 0xb414935c, 0x664465ed, 0x024acac7,
0x59a744c1, 0x1d2936a7, 0xdc580aa6, 0xcf574ca8,
0x040a7a10, 0x6cd81807, 0x8a98be4c, 0xaccea063,
0xc33e92b5, 0xd1e0e03d, 0xb322517e, 0x2092bd13,
0x386b2c4a, 0x52e8dd58, 0x58656dfb, 0x50820371,
0x41811896, 0xe337ef7e, 0xd39fb119, 0xc97f0df6,
0x68fea01b, 0xa150a6e5, 0x55258962, 0xeb6ff41b,
0xd7c9cd7a, 0xa619cd9e, 0xbcf09576, 0x2672c073,
0xf003fb3c, 0x4ab7a50b, 0x1484126a, 0x487ba9b1,
0xa64fc9c6, 0xf6957d49, 0x38b06a75, 0xdd805fcd,
0x63d094cf, 0xf51c999e, 0x1aa4d343, 0xb8495294,
0xce9f8e99, 0xbffcd770, 0xc7c275cc, 0x378453a7,
0x7b21be33, 0x397f41bd, 0x4e94d131, 0x92cc1f98,
0x5915ea51, 0x99f861b7, 0xc9980a88, 0x1d74fd5f,
0xb0a495f8, 0x614deed0, 0xb5778eea, 0x5941792d,
0xfa90c1f8, 0x33f824b4, 0xc4965372, 0x3ff6d550,
0x4ca5fec0, 0x8630e964, 0x5b3fbbd6, 0x7da26a48,
0xb203231a, 0x04297514, 0x2d639306, 0x2eb13149,
0x16a45272, 0x532459a0, 0x8e5f4872, 0xf966c7d9,
0x07128dc0, 0x0d44db62, 0xafc8d52d, 0x06316131,

0xd838e7ce, 0x1bc41d00, 0x3a2e8c0f, 0xea83837e,
0xb984737d, 0x13ba4891, 0xc4f8b949, 0xa6d6acb3,
0xa215cdce, 0x8359838b, 0x6bd1aa31, 0xf579dd52,
0x21b93f93, 0xf5176781, 0x187dfdde, 0xe94aeb76,
0x2b38fd54, 0x431de1da, 0xab394825, 0x9ad3048f,
0xdfea32aa, 0x659473e3, 0x623f7863, 0xf3346c59,
0xab3ab685, 0x3346a90b, 0x6b56443e, 0xc6de01f8,
0x8d421fc0, 0x9b0ed10c, 0x88f1a1e9, 0x54c1f029,
0x7dead57b, 0x8d7ba426, 0x4cf5178a, 0x551a7cca,
0x1a9a5f08, 0xfcd651b9, 0x25605182, 0xe11fc6c3,
0xb6fd9676, 0x337b3027, 0xb7c8eb14, 0x9e5fd030,
0x6b57e354, 0xad913cf7, 0x7e16688d, 0x58872a69,
0x2c2fc7df, 0xe389ccc6, 0x30738df1, 0x0824a734,
0xe1797a8b, 0xa4a8d57b, 0x5b5d193b, 0xc8a8309b,
0x73f9a978, 0x73398d32, 0x0f59573e, 0xe9df2b03,
0xe8a5b6c8, 0x848d0704, 0x98df93c2, 0x720a1dc3,
0x684f259a, 0x943ba848, 0xa6370152, 0x863b5ea3,
0xd17b978b, 0x6d9b58ef, 0x0a700dd4, 0xa73d36bf,
0x8e6a0829, 0x8695bc14, 0xe35b3447, 0x933ac568,
0x8894b022, 0x2f511c27, 0xddfbcc3c, 0x006662b6,
0x117c83fe, 0x4e12b414, 0xc2bca766, 0x3a2fec10,
0xf4562420, 0x55792e2a, 0x46f5d857, 0xceda25ce,
0xc3601d3b, 0x6c00ab46, 0xefac9c28, 0xb3c35047,
0x611dfee3, 0x257c3207, 0xfdd58482, 0x3b14d84f,
0x23becb64, 0xa075f3a3, 0x088f8ead, 0x07adf158,
0x7796943c, 0xfacabf3d, 0xc09730cd, 0xf7679969,
0xda44e9ed, 0x2c854c12, 0x35935fa3, 0x2f057d9f,
0x690624f8, 0x1cb0bafd, 0x7b0dbdc6, 0x810f23bb,
0xfa929a1a, 0x6d969a17, 0x6742979b, 0x74ac7d05,
0x010e65c4, 0x86a3d963, 0xf907b5a0, 0xd0042bd3,
0x158d7d03, 0x287a8255, 0xbba8366f, 0x096edc33,
0x21916a7b, 0x77b56b86, 0x951622f9, 0xa6c5e650,
0x8cea17d1, 0xcd8c62bc, 0xa3d63433, 0x358a68fd,
0x0f9b9d3c, 0xd6aa295b, 0xfe33384a, 0xc000738e,
0xcd67eb2f, 0xe2eb6dc2, 0x97338b02, 0x06c9f246,
0x419cf1ad, 0x2b83c045, 0x3723f18a, 0xcb5b3089,
0x160bead7, 0x5d494656, 0x35f8a74b, 0x1e4e6c9e,
0x000399bd, 0x67466880, 0xb4174831, 0xacf423b2,
0xca815ab3, 0x5a6395e7, 0x302a67c5, 0x8bdb446b,
0x108f8fa4, 0x10223eda, 0x92b8b48b, 0x7f38d0ee,
0xab2701d4, 0x0262d415, 0xaf224a30, 0xb3d88aba,
0xf8b2c3af, 0xdaf7ef70, 0xcc97d3b7, 0xe9614b6c,
0x2baebff4, 0x70f687cf, 0x386c9156, 0xce092ee5,
0x01e87da6, 0x6ce91e6a, 0xbb7bcc84, 0xc7922c20,
0x9d3b71fd, 0x060e41c6, 0xd7590f15, 0x4e03bb47,
0x183c198e, 0x63eeb240, 0x2ddbf49a, 0x6d5cba54,
0x923750af, 0xf9e14236, 0x7838162b, 0x59726c72,
0x81b66760, 0xbb2926c1, 0x48a0ce0d, 0xa6c0496d,
0xad43507b, 0x718d496a, 0x9df057af, 0x44b1bde6,
0x054356dc, 0xde7ced35, 0xd51a138b, 0x62088cc9,
0x35830311, 0xc96efca2, 0x686f86ec, 0x8e77cb68,

```
0x63e1d6b8, 0xc80f9778, 0x79c491fd, 0x1b4c67f2,
0x72698d7d, 0x5e368c31, 0xf7d95e2e, 0xa1d3493f,
0xdcd9433e, 0x896f1552, 0x4bc4ca7a, 0xa6d1baf4,
0xa5a96dcc, 0x0bef8b46, 0xa169fda7, 0x74df40b7,
0x4e208804, 0x9a756607, 0x038e87c8, 0x20211e44,
0x8b7ad4bf, 0xc6403f35, 0x1848e36d, 0x80bdb038,
0x1e62891c, 0x643d2107, 0xbf04d6f8, 0x21092c8c,
0xf644f389, 0x0778404e, 0x7b78adb8, 0xa2c52d53,
0x42157abe, 0xa2253e2e, 0x7bf3f4ae, 0x80f594f9,
0x953194e7, 0x77eb92ed, 0xb3816930, 0xda8d9336,
0xbf447469, 0xf26d9483, 0xee6faed5, 0x71371235,
0xde425f73, 0xb4e59f43, 0x7dbe2d4e, 0x2d37b185,
0x49dc9a63, 0x98c39d98, 0x1301c9a2, 0x389b1bbf,
0x0c18588d, 0xa421c1ba, 0x7aa3865c, 0x71e08558,
0x3c5cfcaa, 0x7d239ca4, 0x0297d9dd, 0xd7dc2830,
0x4b37802b, 0x7428ab54, 0xaeee0347, 0x4b3fbb85,
0x692f2f08, 0x134e578e, 0x36d9e0bf, 0xae8b5fcf,
0xedb93ecf, 0x2b27248e, 0x170eb1ef, 0x7dc57fd6,
0x1e760f16, 0xb1136601, 0x864e1b9b, 0xd7ea7319,
0x3ab871bd, 0xcfa4d76f, 0xe31bd782, 0x0dbeb469,
0xabb96061, 0x5370f85d, 0xffb07e37, 0xda30d0fb,
0xebc977b6, 0x0b98b40f, 0x3a4d0fe6, 0xdf4fc26b,
0x159cf22a, 0xc298d6e2, 0x2b78ef6a, 0x61a94ac0,
0xab561187, 0x14eea0f0, 0xdf0d4164, 0x19af70ee
};

/*
* The following implements Intermediate Values Tests Macros.
* Since internal words are always kept little-endian, always
* swap bytes before displaying.
*/
/*
#ifdef IVT
int ivt_debug = 0;
FILE *ivt_fp;
int ivt_l = 0;
#define IVTSWAP(x) ¥
( (( x                ) << 24) | ¥
(((x)&0xff00    ) << 8 ) | ¥
(((x)&0xff0000) >> 8 ) | ¥
(( x                ) >> 24)    )
#define IVT_DEBUG(a,b,c,d) ¥
if (ivt_debug) ¥
fprintf(ivt_fp,"IV%d=%8.8lx %8.8lx %8.8lx %8.8lx¥n",ivt_l++,¥
IVTSWAP(a),IVTSWAP(b),IVTSWAP(c),IVTSWAP(d));
#else
#define IVT_DEBUG(a,b,c,d)
#endif
*/
/*****************************************************************************
*
```

```
*     Low Level key setup, block encrypt and decrypt routines.
*     For efficiency, these are WORD oriented. The high level NIST
*     routines provide BYTE oriented interfaces, with ENDIAN conversion.
*
*************************************************************************/
/* if multiplication subkey k has 10 0's or 10 1's, mask in a fixing value */
static int fix_subkey(int k, int r)
{
/* the mask words come from S[265]..S[268], as chosen by index.c */
int[] B = S;
int m1, m2;
int i;

i = k & 3;                    /* store the least two bits of k */
k |= 3;                       /* and then mask them away       */

/* we look for 9 consequtive 1's in m1 */
m1 = (~k) ^ (k<<1);     /* for i > 1, m1_i = 1 iff k_i = k_{i-1} */
m2 = m1 & (m1 << 1);    /* m2_i = AND (m1_i, m1_{i-1})     */
m2 &= m2 << 2;          /* m2_i = AND (m1_i...m1_{i-3})    */
m2 &= m2 << 4;          /* m2_i = AND (m1_i...m1_{i-7})    */
m2 &= m1 << 8;          /* m2_i = AND (m1_i...m1_{i-8})    */
m2 &= 0xfffffe00;       /* mask out the low 9 bits of m2 */
/* for i = 9...31, m2_i = 1 iff k_i = ... = k_{i-9} */

/* if m2 is zero, k was good, so return */
if (m2 == 0)
return(k);

/* need to fix k: we copy each 1 in m2 to the nine bits to its right */
m1 = m2 | (m2 >> 1);    /* m1_i = AND (m2_i, m2_{i+1})     */
m1 |= m1 >> 2;          /* m1_i = AND (m2_i...m2_{i+3})    */
m1 |= m2 >> 4;          /* m1_i = AND (m2_i...m2_{i+4})    */
m1 |= m1 >> 5;          /* m1_i = AND (m2_i...m2_{i+9})    */
/* m1_i = 1 iff k_i belongs to a sequence of ten 0's or ten 1's */

/* we turn off the two lowest bits of M, and also every bit
* M_i such that k_i is not equal to both k_{i-1} and k_{i+1}
*/
m1 &= ((~k)^(k<<1)) & ((~k)^(k>>1)) & 0x7ffffffc;

/* and finally pick a pattern, rotate it,
* and xor it into k under the control of the mask m1
*/



k ^=     (((B[i])<<(int)(r)) | ((B[i])>>(32 - (int)(r)))) & m1;

return(k);
}
```

```
/* setup a mars key schedule
*
* n    (input) is the number of words in the key
* kp (input) is a pointer to the array of key words
* ep (output) is a pointer to the array of EKEY_WORDS expanded subkey WORDs
*/
int mars_setup(int n, int[] kp, int[] ep)
{
int[] T = new int[15];// = {0};
int i,j,t;

/* check key length */
if ((n<4)||(n>14))
return(-2);//BAD_KEY_MAT);

/* initialize the T[] array with key data */
for (i=0; i<n; i++)
T[i] = kp[i];
T[n] = n;
for (i=n+1; i<15; i++)
T[i] = 0;

/* Four iterations, each one computing 10 words of the array */
for (j=0; j<4; j++) {
int w;

/* Linear transformation */
w = T[8] ^ T[13]; T[0] ^= (((w)<<(int)(3)) | ((w)>>(32 - (int)(3))))^ j;
w = T[9] ^ T[14]; T[1] ^= (((w)<<(int)(3)) | ((w)>>(32 - (int)(3)))) ^ (4+j);
for (i=2; i<7; i++) {
w = T[i+8] ^ T[i-2];
T[i] ^= (((w)<<(int)(3)) | ((w)>>(32 - (int)(3)))) ^ ((i<<2)+j);
}
for (i=7; i<15; i++) {
w = T[i-7] ^ T[i-2];
T[i] ^= (((w)<<(int)(3)) | ((w)>>(32 - (int)(3)))) ^ ((i<<2)+j);
}

/* Four stirring rounds */
for (t=0; t<4; t++){
/* stir with full type-1 s-box rounds */
T[0] += S[ T[14]&511 ];
T[0] = (((T[0])<<(int)(9)) | ((T[0])>>(32 - (int)(9))));//LROTATE(T[0],9);
for (i=1; i<15; i++) {
T[i] += S[ T[i-1]&511 ];
T[i] = (((T[i])<<(int)(9)) | ((T[i])>>(32 - (int)(9))));//LROTATE(T[i],9);
}
}
```

```
/* copy subkeys to mars_ctx, with swapping around */
/*
#define SWAP(i) (ep[(10*j)+i] = T[(i*4)%15])

SWAP(0); SWAP(1); SWAP(2); SWAP(3); SWAP(4);
SWAP(5); SWAP(6); SWAP(7); SWAP(8); SWAP(9);
*/
ep[(10*j)+0] = T[(0*4)%15];
ep[(10*j)+1] = T[(1*4)%15];
ep[(10*j)+2] = T[(2*4)%15];
ep[(10*j)+3] = T[(3*4)%15];
ep[(10*j)+4] = T[(4*4)%15];
ep[(10*j)+5] = T[(5*4)%15];
ep[(10*j)+6] = T[(6*4)%15];
ep[(10*j)+7] = T[(7*4)%15];
ep[(10*j)+8] = T[(8*4)%15];
ep[(10*j)+9] = T[(9*4)%15];

}


/* IBM specific defines: these parameters can be changed                    */
//#define NUM_MIX 8                 /* number of mixing rounds per stage          */
//#define NUM_ROUNDS 16             /* number of full core rounds                 */
//#define NUM_SETUP 7               /* number of key setup mixing rounds          */

/* IBM specific defines: these parameters are fixed for this implementation */
//#define W                 32          /* number of bits in a word                 */
//#define NUM_DATA      4           /* data block size in words                     */
//#define EKEY_DWORDS (2*(NUM_DATA+NUM_ROUNDS))     /* number of subkey words     */

/* IBM modified values                                                          */
//#define MAX_KEY_SIZE (EKEY_DWORDS*8)   /* max ASCII char's needed for a key     */
//#define MAX_IV_SIZE (NUM_DATA*4)       /* max bytes's needed for an IV          */


/* check and fix all multiplication subkeys */
for (i=4+1;i<((2*(4+16)) - 4);i+=2)
ep[i] = fix_subkey(ep[i], ep[i-1]);

return(1);
}



//#define
void MixForwardRound(int d1, int d2, int d3, int d4,   int[] sp){
            int w,x,y,z;
d2 ^= sp[d1&255];
y =         (((d1)>>(int)(8)) | ((d1)<<(32 - (int)(8))));
z =       (((d1)>>(int)(16)) | ((d1)<<(32 - (int)(16))));
```

```
d1 =      (((d1)>>(int)(24)) | ((d1)<<(32 - (int)(24))));
d2 += sp[(y&255)+256];
d3 += sp[z&255];
d4 ^= sp[(d1&255)+256];
}

//#define
void MixBackwardsRound(int d1, int d2, int d3, int d4, int[] sp) {
            int w,x,y,z;
d2 ^= sp[(d1&255)+256];
y =       (((d1)<<(int)(8)) | ((d1)>>(32 - (int)(8))));
z =       (((d1)<<(int)(16)) | ((d1)>>(32 - (int)(16))));
d1 =      (((d1)<<(int)(24)) | ((d1)>>(32 - (int)(24))));
d3 -= sp[y&255];
d4 -= sp[(z&255)+256];
d4 ^= sp[d1&255];
}

//#define
void CoreRound(int d1, int d2, int d3, int d4,int i, int[] ep, int[] sp){
            int w,x,y,z,tmp;
y = d1;
d1 += ep[i];
y = (((y)<<(int)(13)) | ((y)>>(32 - (int)(13))));
z = d1;
tmp = y;
y *= ep[(i)+1];
z &= 511;
z = sp[z];
y =   (((y)<<(int)(5)) | ((y)>>(32 - (int)(5))));
z ^= y;
d1 = (((d1)<<(int)(y)) | ((d1)>>(32 - (int)(y))));
y =     (((y)<<(int)(5)) | ((y)>>(32 - (int)(5))));
d3 += d1;
z ^= y;
z =     (((z)<<(int)(y)) | ((z)>>(32 - (int)(y))));
d2 += z;
d4 ^= y;
d1 = tmp;
}

//#define
void InvCoreRound(int d1, int d2, int d3, int d4, int i, int[] ep, int[] sp) {
            int w,x,y,z,tmp;
y = d1;
d1 = (((d1)>>(int)(13)) | ((d1)<<(32 - (int)(13))));
y *= ep[(i)+1];
tmp = d1;
d1 += ep[i];
z = d1;
y =       (((y)<<(int)(5)) | ((y)>>(32 - (int)(5))));
```

```
z &= 511;
z = sp[z];
d1 =    (((d1)<<(int)(y)) | ((d1)>>(32 - (int)(y))));
z ^= y;
y =       (((y)<<(int)(5)) | ((y)>>(32 - (int)(5))));
d3 -= d1;
z ^= y;
d1 = tmp;
z =       (((z)<<(int)(y)) | ((z)>>(32 - (int)(y))));
d4 ^= y;
d2 -= z;
}


int NO_MIX = 0;

/* The basic mars encryption: (ep is the expanded key array) */
void mars_encrypt(int[] in, int ins, int[] out, int outs, int[] ep)
{
int a,b,c,d,y,z;
int tmp;
int[] sp = S;

a = in[0];
b = in[1];
c = in[2];
d = in[3];
//IVT_DEBUG(a,b,c,d);


//#ifndef NO_MIX
if(NO_MIX == 0){
/* first, add subkeys to all input data words */
a += ep[0];
b += ep[1];
c += ep[2];
d += ep[3];
//IVT_DEBUG(a,b,c,d);

/* then do eight mixing rounds */
MixForwardRound(a,b,c,d,sp);
a += d;
//IVT_DEBUG(a,b,c,d);
MixForwardRound(b,c,d,a,sp);
b += c;
//IVT_DEBUG(a,b,c,d);
MixForwardRound(c,d,a,b,sp);
//IVT_DEBUG(a,b,c,d);
MixForwardRound(d,a,b,c,sp);
//IVT_DEBUG(a,b,c,d);
```

```
MixForwardRound(a,b,c,d,sp);
a += d;
//IVT_DEBUG(a,b,c,d);
MixForwardRound(b,c,d,a,sp);
b += c;
//IVT_DEBUG(a,b,c,d);
MixForwardRound(c,d,a,b,sp);
//IVT_DEBUG(a,b,c,d);
MixForwardRound(d,a,b,c,sp);
//IVT_DEBUG(a,b,c,d);
}
//#endif

/* then sixteen mars encrypting rounds                    *
* (eight in forward- and eight in backwards-mode) */

CoreRound(a,b,c,d,4,ep,sp);
//IVT_DEBUG(a,b,c,d);
CoreRound(b,c,d,a,6,ep,sp);
//IVT_DEBUG(a,b,c,d);
CoreRound(c,d,a,b,8,ep,sp);
//IVT_DEBUG(a,b,c,d);
CoreRound(d,a,b,c,10,ep,sp);
//IVT_DEBUG(a,b,c,d);

CoreRound(a,b,c,d,12,ep,sp);
//IVT_DEBUG(a,b,c,d);
CoreRound(b,c,d,a,14,ep,sp);
//IVT_DEBUG(a,b,c,d);
CoreRound(c,d,a,b,16,ep,sp);
//IVT_DEBUG(a,b,c,d);
CoreRound(d,a,b,c,18,ep,sp);
//IVT_DEBUG(a,b,c,d);

CoreRound(a,d,c,b,20,ep,sp);
//IVT_DEBUG(a,b,c,d);
CoreRound(b,a,d,c,22,ep,sp);
//IVT_DEBUG(a,b,c,d);
CoreRound(c,b,a,d,24,ep,sp);
//IVT_DEBUG(a,b,c,d);
CoreRound(d,c,b,a,26,ep,sp);
//IVT_DEBUG(a,b,c,d);

CoreRound(a,d,c,b,28,ep,sp);
//IVT_DEBUG(a,b,c,d);
CoreRound(b,a,d,c,30,ep,sp);
//IVT_DEBUG(a,b,c,d);
CoreRound(c,b,a,d,32,ep,sp);
//IVT_DEBUG(a,b,c,d);
CoreRound(d,c,b,a,34,ep,sp);
//IVT_DEBUG(a,b,c,d);
```

```
//#ifndef NO_MIX
if(NO_MIX == 0){
/* then do eight inverse-mixing rounds */
MixBackwardsRound(a,b,c,d,sp);
//IVT_DEBUG(a,b,c,d);
MixBackwardsRound(b,c,d,a,sp);
c -= b;
//IVT_DEBUG(a,b,c,d);
MixBackwardsRound(c,d,a,b,sp);
d -= a;
//IVT_DEBUG(a,b,c,d);
MixBackwardsRound(d,a,b,c,sp);
//IVT_DEBUG(a,b,c,d);

MixBackwardsRound(a,b,c,d,sp);
//IVT_DEBUG(a,b,c,d);
MixBackwardsRound(b,c,d,a,sp);
c -= b;
//IVT_DEBUG(a,b,c,d);
MixBackwardsRound(c,d,a,b,sp);
d -= a;
//IVT_DEBUG(a,b,c,d);
MixBackwardsRound(d,a,b,c,sp);
//IVT_DEBUG(a,b,c,d);

/* subtract final subkeys */
a -= ep[2*16+4];
b -= ep[2*16+5];
c -= ep[2*16+6];
d -= ep[2*16+7];
//IVT_DEBUG(a,b,c,d);
}
//#endif

out[0] = a;
out[1] = b;
out[2] = c;
out[3] = d;
}

/* mars decryption is simply encryption in reverse */
void mars_decrypt(int[] in, int ins, int[] out, int outs, int[] ep)
{
int a,b,c,d,y,z;
int tmp;
int[] sp = S;

d = in[0];
c = in[1];
b = in[2];
```

```
a = in[3];
//IVT_DEBUG(d,c,b,a);

//#ifndef NO_MIX
if(NO_MIX == 0){
/* first, add subkeys to all input data DWORDs */
a += ep[2*16+7];
b += ep[2*16+6];
c += ep[2*16+5];
d += ep[2*16+4];
//IVT_DEBUG(d,c,b,a);

/* then do eight mixing rounds */
MixForwardRound(a,b,c,d,sp);
//IVT_DEBUG(d,c,b,a);
a += d;
MixForwardRound(b,c,d,a,sp);
//IVT_DEBUG(d,c,b,a);
b += c;
MixForwardRound(c,d,a,b,sp);
//IVT_DEBUG(d,c,b,a);
MixForwardRound(d,a,b,c,sp);
//IVT_DEBUG(d,c,b,a);

MixForwardRound(a,b,c,d,sp);
//IVT_DEBUG(d,c,b,a);
a += d;
MixForwardRound(b,c,d,a,sp);
//IVT_DEBUG(d,c,b,a);
b += c;
MixForwardRound(c,d,a,b,sp);
//IVT_DEBUG(d,c,b,a);
MixForwardRound(d,a,b,c,sp);
//IVT_DEBUG(d,c,b,a);
}
//#endif

/* then sixteen mars decrypting rounds                    *
* (eight in forward- and eight in backwards-mode) */

InvCoreRound(a,b,c,d,34,ep,sp);
//IVT_DEBUG(d,c,b,a);
InvCoreRound(b,c,d,a,32,ep,sp);
//IVT_DEBUG(d,c,b,a);
InvCoreRound(c,d,a,b,30,ep,sp);
//IVT_DEBUG(d,c,b,a);
InvCoreRound(d,a,b,c,28,ep,sp);
//IVT_DEBUG(d,c,b,a);

InvCoreRound(a,b,c,d,26,ep,sp);
//IVT_DEBUG(d,c,b,a);
```

```
InvCoreRound(b,c,d,a,24,ep,sp);
//IVT_DEBUG(d,c,b,a);
InvCoreRound(c,d,a,b,22,ep,sp);
//IVT_DEBUG(d,c,b,a);
InvCoreRound(d,a,b,c,20,ep,sp);
//IVT_DEBUG(d,c,b,a);

InvCoreRound(a,d,c,b,18,ep,sp);
//IVT_DEBUG(d,c,b,a);
InvCoreRound(b,a,d,c,16,ep,sp);
//IVT_DEBUG(d,c,b,a);
InvCoreRound(c,b,a,d,14,ep,sp);
//IVT_DEBUG(d,c,b,a);
InvCoreRound(d,c,b,a,12,ep,sp);
//IVT_DEBUG(d,c,b,a);

InvCoreRound(a,d,c,b,10,ep,sp);
//IVT_DEBUG(d,c,b,a);
InvCoreRound(b,a,d,c,8,ep,sp);
//IVT_DEBUG(d,c,b,a);
InvCoreRound(c,b,a,d,6,ep,sp);
//IVT_DEBUG(d,c,b,a);
InvCoreRound(d,c,b,a,4,ep,sp);
//IVT_DEBUG(d,c,b,a);

//#ifndef NO_MIX
if(NO_MIX == 0){
/* then do eight inverse-mixing rounds */
MixBackwardsRound(a,b,c,d,sp);
//IVT_DEBUG(d,c,b,a);
MixBackwardsRound(b,c,d,a,sp);
//IVT_DEBUG(d,c,b,a);
c -= b;
MixBackwardsRound(c,d,a,b,sp);
//IVT_DEBUG(d,c,b,a);
d -= a;
MixBackwardsRound(d,a,b,c,sp);
//IVT_DEBUG(d,c,b,a);

MixBackwardsRound(a,b,c,d,sp);
//IVT_DEBUG(d,c,b,a);
MixBackwardsRound(b,c,d,a,sp);
//IVT_DEBUG(d,c,b,a);
c -= b;
MixBackwardsRound(c,d,a,b,sp);
//IVT_DEBUG(d,c,b,a);
d -= a;
MixBackwardsRound(d,a,b,c,sp);
//IVT_DEBUG(d,c,b,a);

/* subtract final subkeys */
```

```
a -= ep[3];
b -= ep[2];
c -= ep[1];
d -= ep[0];
//IVT_DEBUG(d,c,b,a);
}
//#endif

out[0] = d;
out[1] = c;
out[2] = b;
out[3] = a;
}

/*************************************************************************
*
*     NIST High Level key setup, block encrypt and decrypt routines
*
*************************************************************************/

/* table for rapid, case insensitive hex conversion */
byte[] hex = {
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 0, 0, 0, 0, 0,
0,10,11,12,13,14,15, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,10,11,12,13,14,15, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };



/* NIST defined high level key setup */
int makeKey(MARSkeyInstance key, int direction, int keyLen, int[] keyMaterial)
{
int[] tmpkey = new int[(2*(4+16))];
int i,j;

/* sanity check pointers */
if (key == null)
return(-3);
if (keyMaterial == null)
```

```
return(-2);

/* save parameters into keyInstance */
key.direction = direction;
key.keyLen = keyLen;
for (i=0;i<keyLen/4;i++)
key.keyMaterial[i] = keyMaterial[i];
key.keyMaterial[(2*(4+16))*8] = '¥0';

/* convert ascii keyMaterial to BYTEs */
for(i=0,j=0;i<keyLen/4;i+=2,j++)

(/*(BYTE *)*/tmpkey)[j] = (byte)((hex[(int)keyMaterial[i]]<<4) | hex[(int)keyMaterial[i+1]]);

//#    ifdef SWAP_BYTES

if(SWAP_BYTES == 1){
/* BSWAP the input key DWORDs */
for (i=0;i<keyLen/32;i++)
tmpkey[i] = BSWAP(tmpkey[i]);
}
//#    endif

/* call low level mars setup routine */
return(mars_setup(keyLen/32,tmpkey,key.E));
}

int cipherInit(MARScipherInstance cipher, int mode, int[] IV)
{
int i,j;

/* sanity check pointers */
if (cipher == null)
return(-4);

/* save cipher parameters */
cipher.mode = mode;

/* handle IV */
if((mode == 2)||(mode == 3)) {
if(IV == null)
return(-4);
/* convert ascii IV to BYTEs */
for(i=0,j=0;j<4*4;i+=2,j++)
cipher.IV[j] = (byte)((hex[(int)IV[i]]<<4) | hex[(int)IV[i+1]]);
/* copy BYTE IV to DWORD CIV, with conversion if necessary */
for(i=0;i<4;i++)
cipher.CIV[i] = BSWAP((cipher.IV)[i]);
}
return(1);
}
```

```
/* this assumes the input length is a multiple of 128 bits */
int blockEncrypt(MARScipherInstance cipher, MARSkeyInstance key, int[] input,
                        int inputLen, int[] outBuffer)
{
        int[] tmp = new int[4];
        int i;

        if (cipher.mode == 1) {
                for(i=0;i<inputLen/8;i+=16){
                        if(SWAP_BYTES == 1){
                                tmp[0] = BSWAP(input[i+0]);
                                tmp[1] = BSWAP(input[i+4]);
                                tmp[2] = BSWAP(input[i+8]);
                                tmp[3] = BSWAP(input[i+12]);
                                mars_encrypt(tmp, 0, outBuffer, i, key.E);
                                outBuffer[i+0] = (byte) BSWAP(outBuffer[i+0]);
                                outBuffer[i+4] = (byte) BSWAP(outBuffer[i+4]);
                                outBuffer[i+8] = (byte) BSWAP(outBuffer[i+8]);
                                outBuffer[i+12] = (byte) BSWAP(outBuffer[i+12]);
                        }
                        //#                else
                        else{
                        mars_encrypt(input, i, outBuffer, i, key.E);
                        }
                //#                endif
                }
        }
        else if(cipher.mode == 2) {
                for(i=0;i<inputLen/8;i+=16){
                        //#                ifdef SWAP_BYTES
                        if(SWAP_BYTES == 1){
                                tmp[0] = BSWAP(input[i+0]) ^ cipher.CIV[0];
                                tmp[1] = BSWAP(input[i+4]) ^ cipher.CIV[1];
                                tmp[2] = BSWAP(input[i+8]) ^ cipher.CIV[2];
                                tmp[3] = BSWAP(input[i+12]) ^ cipher.CIV[3];
                                mars_encrypt(tmp, 0, outBuffer, i, key.E);
                                cipher.CIV[0] = outBuffer[i+0];
                                cipher.CIV[1] = outBuffer[i+4];
                                cipher.CIV[2] = outBuffer[i+8];
                                cipher.CIV[3] = outBuffer[i+12];
                                outBuffer[i+0] = (byte) BSWAP(outBuffer[i+0]);
                                outBuffer[i+4] = (byte) BSWAP(outBuffer[i+4]);
                                outBuffer[i+8] = (byte) BSWAP(outBuffer[i+8]);
                                outBuffer[i+12] = (byte) BSWAP(outBuffer[i+12]);
                        }
                        //#
                        else{
                                tmp[0] = input[i+0] ^ cipher.CIV[0];
                                tmp[1] = input[i+4] ^ cipher.CIV[1];
```

```
                                tmp[2] = input[i+8] ^ cipher.CIV[2];
                                tmp[3] = input[i+12] ^ cipher.CIV[3];
                                mars_encrypt(tmp,0,outBuffer,i,key.E);
                                cipher.CIV[0] = outBuffer[i+0];
                                cipher.CIV[1] = outBuffer[i+4];
                                cipher.CIV[2] = outBuffer[i+8];
                                cipher.CIV[3] = outBuffer[i+12];
                        }
                        //#                endif
                }
        }
        else if(cipher.mode == 3) {
                cipher.mode = 1;  /* do encryption in ECB */
                for (int n=0;n<inputLen;n++)
                {
                        blockEncrypt(cipher,key, cipher.IV, 128,    x);
                        bit0    = (byte) (0x80 >> (n & 7));/* which bit position in byte */
                        ctBit = (byte) ((input[n/8] & bit0) ^ (((byte) x[0] & 0x80) >> (n&7)));
                        outBuffer[n/8] = (byte) ((outBuffer[n/8] & ~ bit0) | ctBit);
                        carry = (byte) (ctBit >> (7 - (n&7)));
                        for (i=128/8-1;i>=0;i--)
                        {
                                bit = (byte) (cipher.IV[i] >> 7);       /* save next "carry" from shift */
                                cipher.IV[i] = (byte) ((cipher.IV[i] << 1) ^ carry);
                                carry = bit;
                        }
                }
                cipher.mode = 3;  /* restore mode for next time */
                return inputLen;
                /* Input is one bit (lsb of first byte).
                  * Encrypt IV with key, xor input with msb of encrypted IV,
                  * and then feed output cipher bit into lsb of IV.
                  */
                /*
        DWORD ECIV[4];

        if(inputLen != 1)
        return(BAD_CIPHER_MODE);

        mars_encrypt(cipher->CIV, ECIV, key->E);
        outBuffer[0] = (input[0] & 1)^(ECIV[0]>>31);
        cipher->CIV[0] = (cipher->CIV[0]<<1)|(cipher->CIV[1] & 0x80000000);
        cipher->CIV[1] = (cipher->CIV[1]<<1)|(cipher->CIV[2] & 0x80000000);
        cipher->CIV[2] = (cipher->CIV[2]<<1)|(cipher->CIV[3] & 0x80000000);
        cipher->CIV[3] = (cipher->CIV[3]<<1)|(DWORD)outBuffer[0];*/
        }
        else
                return(-4);

        return(inputLen);
}
```

```
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
int MarsEC(String keyfn, String ptfn, String ctfn)                    // 引数へのポインタ
{
        File fkey, fin, fout;
        int i;
        byte[] c_mode = new byte[3];
        byte[] c_klen = new byte[5];
        int[] c_key = new int[128];
        byte[] c_keyb = null;//new byte[64+2];
        int[] c_cini = new int[64];
        byte[] c_cinib = new byte[32+2];
        int j,k;

        int len, rlen, blen4, pfilelen;
        int mode,klen,blen,rc=0;

        blen4 = 2048;

        MARScipherInstance cipherI = new MARScipherInstance();
        MARSkeyInstance keyI = new MARSkeyInstance();
        ////////////////////////////////

        fkey = new File(keyfn);
        fkey.getParentFile().mkdir();
        FileInputStream inkeyst=null;
        try {
                inkeyst = new FileInputStream(fkey);
                inkeyst.read(c_mode);
                inkeyst.read(c_klen);
                klen = atoi(c_klen);
                c_keyb = new byte[klen/4+2];
                inkeyst.read(c_keyb);
                inkeyst.read(c_cinib);
        } catch (IOException e5) {
                // TODO 自動生成された catch ブロック
                e5.printStackTrace();
        }

        fin = new File(ptfn);
        fin.getParentFile().mkdir();
        FileInputStream inptst=null;
        try {
                inptst = new FileInputStream(fin);
        } catch (FileNotFoundException e5) {
                // TODO 自動生成された catch ブロック
                e5.printStackTrace();
        }
```

```
fout = new File(ctfn);
fout.getParentFile().mkdir();
FileOutputStream outctst=null;
try {
		outctst = new FileOutputStream(fout);
} catch (FileNotFoundException e5) {
		// TODO 自動生成された catch ブロック
		e5.printStackTrace();
}

mode = atoi(c_mode);
klen = atoi(c_klen);
blen = 128;

if(klen<56 || 448<klen){
		//printf("Wrong key size. ¥n");
		return (-1);
}

for(i =0; i<klen/4 ; i++){
		int tmpi = c_keyb[i];
		if(tmpi < 0){
			tmpi = c_keyb[i] ^ 0x80;
			tmpi += 0x80;
		}
		c_key[i] = tmpi;
}
for(i =0; i<32 ; i++){
		int tmpi = c_cinib[i];
		if(tmpi < 0){
			tmpi = c_cinib[i] ^ 0x80;
			tmpi += 0x80;
		}
		c_cini[i] = tmpi;
}

/*Set mode*/
if(mode == 1){
		int[] tmpb = new int[1];
		tmpb[0] = ' ';
		rc=cipherInit(cipherI, 1, tmpb);
}
if(mode == 2){
		rc=cipherInit(cipherI, 2, c_cini);
}
if(mode == 3){
		rc=cipherInit(cipherI, 3, c_cini);
}
if(rc<=0){
		//printf("モード設定が出来ません。");
		return(-2);
```

```
}

makeKey(keyI, 0,    klen, c_key );

int flen;
try {
        flen = inptst.available();
        rlen = flen;

        // reset to start

        s = 4;//sizeof(unsigned int);
        // write the bytes of the file
        //*((unsigned int*)pbuf) = rlen;
        pbufb[0] = (byte) flen;
        pbufb[1] = (byte)(flen>>>8);
        pbufb[2] = (byte)(flen>>>16);
        pbufb[3] = (byte)(flen>>>24);

        if(flen > blen4-s){
                len = inptst.read(pbufb, 4, blen4-s );
        }else{
                len = inptst.read(pbufb, 4, flen);
        }
        rlen -= len;
        int jj =0;
        for(i=0; i*4<len+4; i++){
                for( k=0;k<4;k++){
                        int tmp = pbufb[i*4+3-k];
                        if(tmp < 0){
                                byte tmpb = (byte) (pbufb[i*4+3-k] ^ 0x80);
                                tmp = tmpb;
                                tmp += 0x80;
                        }
                        jj = (jj << 8) | tmp;
                }
                pbuf[i] = jj;
                jj = 0;
        }

        int mesLength = len + 4;
        int block = mesLength/16 + 1;

        oip = 0;
        rc=blockEncrypt(cipherI, keyI, pbuf, 8*blen4, cbuf);
        outctst.write(cbufb, 0, blen4);

        while(rlen > 0 && inptst.available()>0){
                // read a block and reduce the remaining byte count
                len = inptst.read(pbufb, 0, blen4);
                rlen -= len;
```

```
                    rc=blockEncrypt(cipherI, keyI, pbuf, 8*blen4, cbuf);
                    outctst.write(cbufb, 0, blen4);
                }
                if(inkeyst != null)
                {
                    try {
                        inkeyst.close();
                    } catch (IOException e) {
                        // TODO 自動生成された catch ブロック
                        e.printStackTrace();
                    }
                }

                if(outctst != null)
                {
                    try {
                        outctst.close();
                    } catch (IOException e) {
                        // TODO 自動生成された catch ブロック
                        e.printStackTrace();
                    }
                }

                if(inptst != null)
                {
                    try {
                        inptst.close();
                    } catch (IOException e) {
                        // TODO 自動生成された catch ブロック
                        e.printStackTrace();
                    }
                }
            } catch (IOException e1) {
                // TODO 自動生成された catch ブロック
                e1.printStackTrace();
            }
            return 0;
            }


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




/* this assumes the input length is a multiple of 128 bits */
int blockDecrypt(MARScipherInstance cipher, MARSkeyInstance key, int[] input,int inputLen, int[]
outBuffer)
{
        int i;
```

```
        if (cipher.mode == 1) {
                for(i=0;i<inputLen/8;i+=16){
                        //#
                        if( SWAP_BYTES == 1){
                                int[] tmp = new int[4];
                                tmp[0] = BSWAP(input[i+0]);
                                tmp[1] = BSWAP(input[i+4]);
                                tmp[2] = BSWAP(input[i+8]);
                                tmp[3] = BSWAP(input[i+12]);
                                mars_decrypt(tmp,0,outBuffer, i, key.E);
                                outBuffer[i+0] = (byte) BSWAP(outBuffer[i+0]);
                                outBuffer[i+4] = (byte) BSWAP(outBuffer[i+4]);
                                outBuffer[i+8] = (byte) BSWAP(outBuffer[i+8]);
                                outBuffer[i+12] = (byte) BSWAP(outBuffer[+i+12]);
                        }       else{
                                mars_decrypt(input, i, outBuffer, i, key.E);
                        }
                        //#                endif
                }
        }
        else if(cipher.mode == 2) {
                for(i=0;i<inputLen/8;i+=16){
//#             ifdef SWAP_BYTES
                        if(SWAP_BYTES == 1){
                                int[] tmp = new int[4];
                                tmp[0] = BSWAP(input[i+0]);
                                tmp[1] = BSWAP(input[i+4]);
                                tmp[2] = BSWAP(input[i+8]);
                                tmp[3] = BSWAP(input[i+12]);
                                mars_decrypt(tmp,0,outBuffer,i,key.E);
                                outBuffer[i+0] = (byte) BSWAP(outBuffer[i+0] ^ cipher.CIV[0]);
                                outBuffer[i+4] = (byte) BSWAP(outBuffer[i+4] ^ cipher.CIV[1]);
                                outBuffer[i+8] = (byte) BSWAP(outBuffer[i+8] ^ cipher.CIV[2]);
                                outBuffer[i+12] = (byte) BSWAP(outBuffer[i+12] ^ cipher.CIV[3]);
                                cipher.CIV[0] = tmp[0];
                                cipher.CIV[1] = tmp[1];
                                cipher.CIV[2] = tmp[2];
                                cipher.CIV[3] = tmp[3];
                        }
                        //#
                        else{
                                mars_decrypt(input, i, outBuffer, i,key.E);
                                outBuffer[i+0] ^= cipher.CIV[0];
                                outBuffer[i+4] ^= cipher.CIV[1];
                                outBuffer[i+8] ^= cipher.CIV[2];
                                outBuffer[i+12] ^= cipher.CIV[3];
                                cipher.CIV[0] = input[i+0];
                                cipher.CIV[1] = input[i+4];
                                cipher.CIV[2] = input[i+8];
                                cipher.CIV[3] = input[i+12];
```

```
                        }
                        //#                endif

                }
        }
        else if(cipher.mode == 3){
                cipher.mode = 1;  /* do encryption in ECB */
                for (n=0;n<inputLen;n++)
                {
                        blockEncrypt(cipher,key,cipher.IV,128,x);
                        bit0    = (byte) (0x80 >> (n & 7));
                        ctBit = (byte) (input[n/8] & bit0);
                        outBuffer[n/8] = (byte) ((outBuffer[n/8] & ~ bit0) |
                                        (ctBit ^ (((byte) x[0] & 0x80) >> (n&7))));
                        carry = (byte) (ctBit >> (7 - (n&7)));
                        for (i=128/8-1;i>=0;i--)
                        {
                                bit = (byte) (cipher.IV[i] >> 7);       /* save next "carry" from shift */
                                cipher.IV[i] = (byte) ((cipher.IV[i] << 1) ^ carry);
                                carry = bit;
                        }
                }
                cipher.mode = 3;  /* restore mode for next time */
                return inputLen;


/* Input is one bit (lsb of first byte).
* Encrypt IV with key, xor input with msb of encrypted IV,
* and then feed input cipher bit into lsb of IV.
*/
/*
DWORD ECIV[4];

if(inputLen != 1)
return(BAD_CIPHER_MODE);

mars_encrypt(cipher->CIV, ECIV, key->E);
outBuffer[0] = (input[0] & 1)^(ECIV[0]>>31);
cipher->CIV[0] = (cipher->CIV[0]<<1) | (cipher->CIV[1] & 0x80000000);
cipher->CIV[1] = (cipher->CIV[1]<<1) | (cipher->CIV[2] & 0x80000000);
cipher->CIV[2] = (cipher->CIV[2]<<1) | (cipher->CIV[3] & 0x80000000);
cipher->CIV[3] = (cipher->CIV[3]<<1) | (DWORD)(input[0]&1);
*/
        }
        else
                return(-4);

        return(inputLen);
}
```

```
//MarsDC.cpp : コンソール アプリケーション用のエントリ ポイントの定義
//

//#define file_len(x) (unsigned long)x


int atoi( byte s[] ) {
	int i, n, sign;

	for( i = 0; s[i] == ' ' ; i++ )  //先頭の空白を読み飛ばす
		;
	sign = ( s[i] == '-' ) ? -1 : 1;	//符号を保存する
	if( s[i] == '-' || s[i] == '+' )	//符号を飛ばす
		i++;
	for( n = 0; i< s.length - 2 ; i++ )  //s[i]が数字のあいだ、n へ
		n = 10 * n + ( s[i] - '0' );
	return sign * n;				//符号を反映
}

////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
//int main(int argc, char* argv[])
void MarsDC(String keyfn, String ctfn, String ptfn)
{
	int i;
	byte[] c_mode = new byte[3];
	byte[] c_klen = new byte[5];
	int[] c_key = new int[64+2];
	byte[] c_keyb = null;//new byte[64+2];
	int[] c_cini = new int[32+2];
	byte[] c_cinib = new byte[32+2];

	int j,k;

    int len, rlen, blen4, pfilelen;
	int mode,klen,blen,rc=0;

	blen4 = 2048;

	MARScipherInstance cipherI = new MARScipherInstance();
	MARSkeyInstance keyI = new MARSkeyInstance();
	////////////////////////////////////////////////

	try{
	FileOutputStream foutst = openFileOutput(ptfn,MODE_PRIVATE);
	FileInputStream finst = openFileInput(ctfn);

	File fkey = new File(keyfn);
	fkey.getParentFile().mkdir();
	FileInputStream inkeyst=null;
```

```
		try {
			inkeyst = new FileInputStream(fkey);
			inkeyst.read(c_mode);
			inkeyst.read(c_klen);
			klen = atoi(c_klen);
			c_keyb = new byte[klen/4+2];
			inkeyst.read(c_keyb);
			inkeyst.read(c_cinib);
		} catch (IOException e3) {
			// TODO 自動生成された catch ブロック
			e3.printStackTrace();
		}//127

		mode = atoi(c_mode);
		klen = atoi(c_klen);
		blen = 128;

		for(i=0; i<32 ; i++){
			c_cini[i] = c_cinib[i];
		}

		if(klen<56 || 256<klen){
//			printf("Wrong key size. ¥n");
			return;// -1;
		}

		/*Set mode*/
		if(mode == 1){
			int[] tmpb = new int[1];
			tmpb[0] = ' ';
			rc=cipherInit(cipherI, 1, tmpb);
		}
		if(mode == 2){
			rc=cipherInit(cipherI, 2, c_cini);
		}
		if(mode == 3){
			rc=cipherInit(cipherI, 3, c_cini);
		}
		if(rc<=0){
//			printf("モード設定が出来ません。");
			return;// -1;
		}

		for(i=0; i<klen/4;i++){
			c_key[i] = c_keyb[i];
		}


		makeKey(keyI, 1,   klen, c_key );

		 int flen = finst.available();
```

```
rlen = flen;

        s = 4;//sizeof(unsigned long);
        // write the bytes of the file
        if(blen4<=flen){
                len = finst.read(cbufb, 0, blen4 );
        }else{
                len = finst.read(cbufb, 0, flen );
        }
        rlen = rlen - len;

        if(len < blen4){ return ;    }

        int jj =0;
        for(i=0; i*4<len; i++){
                for(int p=0; p<4; p++){
                        int tmp = cbufb[i*4+3-p];
                        if(tmp < 0){
                                byte tmpb = (byte) (cbufb[i*4+3-p] ^ 0x80);
                                tmp = tmpb;
                                tmp += 0x80;
                        }
                        jj = (jj << 8) | tmp;
                }
                cbuf[i] = jj;
                jj = 0;
        }

        ob=0; oip=0;
        rc=blockDecrypt(cipherI, keyI,   cbuf, 8*blen4,   pbuf);

        // 復号文出力
//      pfilelen = *((long*)(pbuf));
        byte[] tmpch4 = new byte[4];
        tmpch4[0] = pbufb[0];//(byte) pbuf[0];
        tmpch4[1] = pbufb[1];//(byte) (pbuf[0]>>>8);
        tmpch4[2] = pbufb[2];//(byte) (pbuf[0]>>>16);
        tmpch4[3] = pbufb[3];//(byte) (pbuf[0]>>>24);
        jj = 0;
        int tmp = 0;
        for (int p = 0; p < tmpch4.length; p++) {
                tmp = (tmpch4[3-p] & 0xff);
                if(tmp < 0){
                        tmpch4[3-p] ^= 0x80;
                        tmp = tmpch4[3-p];
                        tmp += 0x80;
                }
                jj = (jj << 8) | tmp;
        }
        pfilelen = jj;
```

```
	if(pfilelen <= blen4 - s){
		foutst.write(pbufb, s, pfilelen);
		if(inkeyst != null)
		{
			try {
				inkeyst.close();
			} catch (IOException e) {
				// TODO 自動生成された catch ブロック
				e.printStackTrace();
			}
		}

		if(foutst != null)
		{
			try {
				foutst.close();
			} catch (IOException e) {
				// TODO 自動生成された catch ブロック
				e.printStackTrace();
			}
		}

		if(finst != null)
		{
			try {
				finst.close();
			} catch (IOException e) {
				// TODO 自動生成された catch ブロック
				e.printStackTrace();
			}
		}
		return;// 0;
	}
	else{
		foutst.write(pbufb, s, blen4 - s);
		pfilelen -= (blen4 - s);
	}


if((rlen <= blen4) && (rlen > 0))
		{	// if the file length is less than or equal to 2048 bytes
		len = finst.read(cbufb, 1, blen4 );
		rlen -= len;
		if(rlen > 0){ return ; }
			rc=blockDecrypt(cipherI, keyI, cbuf, 8*blen4, pbuf);
			foutst.write(pbufb, 1, pfilelen) ;
			if(inkeyst != null)
			{
				try {
					inkeyst.close();
				} catch (IOException e) {
```

```
                                        // TODO 自動生成された catch ブロック
                                        e.printStackTrace();
                                }
                        }

                        if(foutst != null)
                        {
                                try {
                                        foutst.close();
                                } catch (IOException e) {
                                        // TODO 自動生成された catch ブロック
                                        e.printStackTrace();
                                }
                        }

                        if(finst != null)
                        {
                                try {
                                        finst.close();
                                } catch (IOException e) {
                                        // TODO 自動生成された catch ブロック
                                        e.printStackTrace();
                                }
                        }
                        return ;
                }
                else
                {    // if the file length is more 1024 bytes
                     // read the file a block at a time
                     while(rlen > 0 && finst.available()>0)
                     {
                          // read a block and reduce the remaining byte count
                          len = finst.read(cbufb, 1, blen4);
                          rlen -= len;
                          if((rlen>0) && (len==blen4)){
                                        rc=blockDecrypt(cipherI,  keyI,  cbuf,  8*blen4,
pbuf);
                                        foutst.write(pbufb, 1, blen4);
                                        pfilelen -= blen4;
                                }
                          if(rlen<=0){
                                        rc=blockDecrypt(cipherI,  keyI,  cbuf,  8*blen4,
pbuf);
                                        foutst.write(pbufb, 1, pfilelen);
                                        if(inkeyst != null)
                                        {
                                                try {
                                                        inkeyst.close();
                                                } catch (IOException e) {
                                                        // TODO 自動生成された catch
ブロック
```

```
                                        e.printStackTrace();
                                }
                        }

                        if(foutst != null)
                        {
                                try {
                                        foutst.close();
                                } catch (IOException e) {
                                        // TODO 自動生成された catch
ブロック
                                        e.printStackTrace();
                                }
                        }

                        if(finst != null)
                        {
                                try {
                                        finst.close();
                                } catch (IOException e) {
                                        // TODO 自動生成された catch
ブロック
                                        e.printStackTrace();
                                }
                        }
                        return ;
                }
        }
}
        if(inkeyst != null)
        {
                try {
                        inkeyst.close();
                } catch (IOException e) {
                        // TODO 自動生成された catch ブロック
                        e.printStackTrace();
                }
        }

        if(foutst != null)
        {
                try {
                        foutst.close();
                } catch (IOException e) {
                        // TODO 自動生成された catch ブロック
                        e.printStackTrace();
                }
        }

        if(finst != null)
        {
```

```
                                try {
                                        finst.close();
                                } catch (IOException e) {
                                        // TODO 自動生成された catch ブロック
                                        e.printStackTrace();
                                }
                        }

                        return ;

                }catch (IOException e) {
                        // TODO 自動生成された catch ブロック
                        e.printStackTrace();
                }
        }



}

package yu.com.pcs.jp.sumaho.cg5mail;

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

import yu.com.pcs.jp.sumaho.cg5mail.MailViewActivity.cipherInstance;
import yu.com.pcs.jp.sumaho.cg5mail.MailViewActivity.keyInstance;

import android.app.Activity;
import android.os.Bundle;

public class MistyActivity extends Activity{
        String prgn;
        String keyfn;
        String inputfn;
        String outputfn;

        @Override
        protected void onCreate(Bundle savedInstanceState) {
          super.onCreate(savedInstanceState);
                prgn = "";
            keyfn = "";
            inputfn = "";
            outputfn = "";
        }


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

```
/***
 * MistyEC.h
 * Misty 暗号関数
 */

// 戻り値
int NOERROR =   0;                          // エラーなし
int NOTENOUGHMEMORY =    -1;                                // メモリー不足
int       ACCESSERROR =       -2;                          // アクセスエラー
int       MATHERROR   =       -3;                          // 数学的な誤り
int       KEYLENGTHERROR=  -4;                             // 鍵長不正
int OTHERERROR        =       -5;                          // その他のエラー

// 定数
int MINKEYLENGTH =  32;                             // 最低鍵長（ビット）

////////////////////////////////////////////////////////////////////////
// Misty1.cpp : コンソール アプリケーション用のエントリ ポイントの定義
//
int[] lt = new int[3];
int[] lr = new int[4];
int ts = 0;
int[][] EXTKEY = new int[4][8];

int[] S7 = {
        27,50,51,90,59,16,23,84,91,26,114,115,107,44,102,73,
        31,36,19,108,55,46,63,74,93,15,64,86,37,81,26,4,
        11,70,32,13,123,53,68,66,43,30,65,20,75,121,21,111,
        14,85,9,54,116,12,103,83,40,10,126,56,2,7,96,41,
        25,18,101,47,48,57,8,104,95,120,42,76,100,69,117,61,
        89,72,3,87,124,79,98,60,29,33,94,39,106,112,77,58,
        1,109,110,99,24,119,35,5,38,118,0,49,45,122,127,97,
        80,34,17,6,71,22,82,78,113,62,105,67,52,92,88,125};

int[] S9 = {
        451,203,339,415,483,233,251,53,385,185,279,491,307,9,45,211,
        199,330,55,126,235,356,403,472,163,286,85,44,29,418,355,280,
        331,338,466,15,43,48,314,229,273,312,398,99,227,200,500,27,
        1,157,248,416,365,499,28,326,125,209,130,490,387,301,244,414,
        467,221,482,296,480,236,89,145,17,303,38,220,176,396,271,503,
        231,364,182,249,216,337,257,332,259,184,340,299,430,23,113,12,
        71,88,127,420,308,297,132,349,413,434,419,72,124,81,458,35,
        317,423,357,59,66,218,402,206,193,107,159,497,300,388,250,406,
        481,361,381,49,384,266,148,474,390,318,284,96,373,463,103,281,
        101,104,153,336,8,7,380,183,36,25,222,295,219,228,425,82,
        265,144,412,449,40,435,309,362,374,223,485,392,197,366,478,433,
        195,479,54,238,494,240,147,73,154,438,105,129,293,11,94,180,
        329,455,372,62,315,439,142,454,174,16,149,495,78,242,509,133,
        253,246,160,367,131,138,342,155,316,263,359,152,464,489,3,510,
        189,290,137,210,399,18,51,106,322,237,368,283,226,335,344,305,
```

```
        327,93,275,461,121,353,421,377,158,436,204,34,306,26,232,4,
        391,493,407,57,447,471,39,395,198,156,208,334,108,52,498,110,
        202,37,186,401,254,19,262,47,429,370,475,192,267,470,245,492,
        269,118,276,427,117,268,484,345,84,287,75,196,446,247,41,164,
        14,496,119,77,378,134,139,179,369,191,270,260,151,347,352,360,
        215,187,102,462,252,146,453,111,22,74,161,313,175,241,400,10,
        426,323,379,86,397,358,212,507,333,404,410,135,504,291,167,440,
        321,60,505,320,42,341,282,417,408,213,294,431,97,302,343,476,
        114,394,170,150,277,239,69,123,141,325,83,95,376,178,46,32,
        469,63,457,487,428,68,56,20,177,363,171,181,90,386,456,468,
        24,375,100,207,109,256,409,304,346,5,288,443,445,224,79,214,
        319,452,298,21,6,255,411,166,67,136,80,351,488,289,115,382,
        188,194,201,371,393,501,116,460,486,424,405,31,65,13,442,50,
        61,465,128,168,87,441,354,328,217,261,98,122,33,511,274,264,
        448,169,285,432,422,205,243,92,258,91,473,324,502,173,165,58,
        459,310,383,70,225,30,477,230,311,506,389,140,143,64,437,190,
        120,0,172,272,350,292,2,444,162,234,112,508,278,348,76,450 };

void FL_enc(int k, int[] lr, int r0, int r1, int r2, int r3) {
        lr[r1] ^= lr[r0] & EXTKEY[0][k];
        lr[r3] ^= lr[r2] & EXTKEY[1][(k+2)&7];
        lr[r0] ^= lr[r1] | EXTKEY[1][(k+6)&7];
        lr[r2] ^= lr[r3] | EXTKEY[0][(k+4)&7];
        }

void FL_dec(int k,int[] lr, int r0, int r1, int r2, int r3) {
        lr[r0] ^= lr[r1] | EXTKEY[0][(k+4)&7];
        lr[r2] ^= lr[r3] | EXTKEY[1][(k+6)&7];
        lr[r1] ^= lr[r0] & EXTKEY[1][(k+2)&7];
        lr[r3] ^= lr[r2] & EXTKEY[0][k];
        }

void FI_key(int k, int[] lr, int r0, int r1) {
        lr[r0] = EXTKEY[0][k] >>> 7;
        lr[r1] = EXTKEY[0][k] & 0x7f;
        lr[r0] = S9[lr[r0]] ^ lr[r1];
        lr[r1] = S7[lr[r1]] ^ (lr[r0] & 0x7f);
        lr[r1] ^= EXTKEY[0][(k+1)&7] >>> 9;
        lr[r0] ^= EXTKEY[0][(k+1)&7] & 0x1ff;
        lr[r0] = S9[lr[r0]] ^ lr[r1];
        EXTKEY[3][k] = lr[r1];
        EXTKEY[2][k] = lr[r0];
        EXTKEY[1][k] = lr[r1] << 9 ^ lr[r0];
        }

void FI_txt(int[] lt, int a0, int a1, int k) {
        lt[a1] = lt[a0] >>> 7;
        lt[a0] &= 0x7f;
        lt[a1] = S9[lt[a1]] ^ lt[a0];
        lt[a0] = S7[lt[a0]] ^ lt[a1];
        lt[a1] ^= EXTKEY[2][k];
```

```
        lt[a0] ^= EXTKEY[3][k];
        lt[a0] &= 0x7f;
        lt[a1] = S9[lt[a1]] ^ lt[a0];
        lt[a1] ^= lt[a0] << 9;
        }

void FO_txt(int[] lr, int a0, int a1, int a2, int a3, int k, int[] lt, int t0, int t1, int t2) {
        lt[t0] = lr[a0] ^ EXTKEY[0][k];
        FI_txt(lt, t0, t1, (k+5)&7);
        lt[t1] ^= lr[a1];
        lt[t2] = lr[a1] ^ EXTKEY[0][(k+2)&7];
        FI_txt(lt, t2, t0, (k+1)&7);
        lt[t0] ^= lt[t1];
        lt[t1] ^= EXTKEY[0][(k+7)&7];
        FI_txt(lt, t1, t2, (k+3)&7);
        lt[t2] ^= lt[t0];
        lt[t0] ^= EXTKEY[0][(k+4)&7];
        lr[a2] ^= lt[t0];
        lr[a3] ^= lt[t2];
        }


void misty1(byte[] text, byte[] key, int block, int mode)
{
        int i1, i2;
        byte b1, b2;

        b1 = key[0]; b2 = key[1];
        if(b1>=0){ i1 = b1;}
        else{b1 ^= 0x80; i1 = b1; i1 += 0x80;}
        if(b2>=0){ i2 = b2;}
        else{b2 ^= 0x80; i2 = b2; i2 += 0x80;}
        EXTKEY[0][0] = (i1<<8) ^ i2;

        b1 = key[2]; b2 = key[3];
        if(b1>=0){ i1 = b1;}
        else{b1 ^= 0x80; i1 = b1; i1 += 0x80;}
        if(b2>=0){ i2 = b2;}
        else{b2 ^= 0x80; i2 = b2; i2 += 0x80;}
        EXTKEY[0][1] = (i1<<8) ^ i2;

        b1 = key[4]; b2 = key[5];
        if(b1>=0){ i1 = b1;}
        else{b1 ^= 0x80; i1 = b1; i1 += 0x80;}
        if(b2>=0){ i2 = b2;}
        else{b2 ^= 0x80; i2 = b2; i2 += 0x80;}
        EXTKEY[0][2] = (i1<<8) ^ i2;

        b1 = key[6]; b2 = key[7];
        if(b1>=0){ i1 = b1;}
        else{b1 ^= 0x80; i1 = b1; i1 += 0x80;}
```

```
if(b2>=0){ i2 = b2;}
else{b2 ^= 0x80; i2 = b2; i2 += 0x80;}
EXTKEY[0][3] = (i1<<8) ^ i2;

b1 = key[8]; b2 = key[9];
if(b1>=0){ i1 = b1;}
else{b1 ^= 0x80; i1 = b1; i1 += 0x80;}
if(b2>=0){ i2 = b2;}
else{b2 ^= 0x80; i2 = b2; i2 += 0x80;}
EXTKEY[0][4] = (i1<<8) ^ i2;

b1 = key[10]; b2 = key[11];
if(b1>=0){ i1 = b1;}
else{b1 ^= 0x80; i1 = b1; i1 += 0x80;}
if(b2>=0){ i2 = b2;}
else{b2 ^= 0x80; i2 = b2; i2 += 0x80;}
EXTKEY[0][5] = (i1<<8) ^ i2;

b1 = key[12]; b2 = key[13];
if(b1>=0){ i1 = b1;}
else{b1 ^= 0x80; i1 = b1; i1 += 0x80;}
if(b2>=0){ i2 = b2;}
else{b2 ^= 0x80; i2 = b2; i2 += 0x80;}
EXTKEY[0][6] = (i1<<8) ^ i2;

b1 = key[14]; b2 = key[15];
if(b1>=0){ i1 = b1;}
else{b1 ^= 0x80; i1 = b1; i1 += 0x80;}
if(b2>=0){ i2 = b2;}
else{b2 ^= 0x80; i2 = b2; i2 += 0x80;}
EXTKEY[0][7] = (i1<<8) ^ i2;

FI_key(0,lr,0,1);
FI_key(1,lr,0,1);
FI_key(2,lr,0,1);
FI_key(3,lr,0,1);
FI_key(4,lr,0,1);
FI_key(5,lr,0,1);
FI_key(6,lr,0,1);
FI_key(7,lr,0,1);

if((mode & 1) == 0){
        while(block-- > 0){
                b1 = text[ts+0]; b2 = text[ts+1];
                if(b1>=0){ i1 = b1;}
                else{b1 ^= 0x80; i1 = b1; i1 += 0x80;}
                if(b2>=0){ i2 = b2;}
                else{b2 ^= 0x80; i2 = b2; i2 += 0x80;}
                lr[0] = (i1<<8) ^ i2;

                b1 = text[ts+2]; b2 = text[ts+3];
```

```
                    if(b1>=0){ i1 = b1;}
                    else{b1 ^= 0x80; i1 = b1; i1 += 0x80;}
                    if(b2>=0){ i2 = b2;}
                    else{b2 ^= 0x80; i2 = b2; i2 += 0x80;}
                    lr[1] = (i1<<8) ^ i2;

                    b1 = text[ts+4]; b2 = text[ts+5];
                    if(b1>=0){ i1 = b1;}
                    else{b1 ^= 0x80; i1 = b1; i1 += 0x80;}
                    if(b2>=0){ i2 = b2;}
                    else{b2 ^= 0x80; i2 = b2; i2 += 0x80;}
                    lr[2] = (i1<<8) ^ i2;

                    b1 = text[ts+6]; b2 = text[ts+7];
                    if(b1>=0){ i1 = b1;}
                    else{b1 ^= 0x80; i1 = b1; i1 += 0x80;}
                    if(b2>=0){ i2 = b2;}
                    else{b2 ^= 0x80; i2 = b2; i2 += 0x80;}
                    lr[3] = (i1<<8) ^ i2;

                    FL_enc(0,lr, 0,1,2,3);
                    FO_txt(lr, 0, 1, 2, 3, 0, lt, 0, 1, 2);
                    FO_txt(lr, 2, 3, 0, 1, 1, lt, 0, 1, 2);
                    FL_enc(1,lr, 0, 1, 2, 3);
                    FO_txt(lr, 0, 1, 2, 3, 2, lt, 0, 1, 2);
                    FO_txt(lr, 2, 3, 0, 1, 3,lt, 0, 1, 2);
                    FL_enc(2,lr, 0, 1, 2, 3);
                    FO_txt(lr, 0, 1, 2, 3, 4, lt, 0, 1, 2);
                    FO_txt(lr, 2, 3, 0, 1, 5, lt, 0, 1, 2);
                    FL_enc(3,lr, 0, 1, 2, 3);
                    FO_txt(lr, 0, 1, 2, 3, 6, lt, 0, 1, 2);
                    FO_txt(lr, 2, 3, 0, 1, 7, lt, 0, 1, 2);
                    FL_enc(4, lr, 0, 1, 2, 3);

                    text[ts+0] = (byte) (lr[2] >>> 8);
                    text[ts+1] = (byte) (lr[2] & 0xff);
                    text[ts+2] = (byte) (lr[3] >>> 8);
                    text[ts+3] = (byte) (lr[3] & 0xff);
                    text[ts+4] = (byte) (lr[0] >>> 8);
                    text[ts+5] = (byte) (lr[0] & 0xff);
                    text[ts+6] = (byte) (lr[1] >>> 8);
                    text[ts+7] = (byte) (lr[1] & 0xff);

                    ts += 8;
            }
    }
    else{
            while(block-- > 0){
                    b1 = text[ts+0]; b2 = text[ts+1];
                    if(b1>=0){ i1 = b1;}
                    else{b1 ^= 0x80; i1 = b1; i1 += 0x80;}
```

```
                    if(b2>=0){ i2 = b2;}
                    else{b2 ^= 0x80; i2 = b2; i2 += 0x80;}
                    lr[0] = (i1<<8) ^ i2;

                    b1 = text[ts+2]; b2 = text[ts+3];
                    if(b1>=0){ i1 = b1;}
                    else{b1 ^= 0x80; i1 = b1; i1 += 0x80;}
                    if(b2>=0){ i2 = b2;}
                    else{b2 ^= 0x80; i2 = b2; i2 += 0x80;}
                    lr[1] = (i1<<8) ^ i2;

                    b1 = text[ts+4]; b2 = text[ts+5];
                    if(b1>=0){ i1 = b1;}
                    else{b1 ^= 0x80; i1 = b1; i1 += 0x80;}
                    if(b2>=0){ i2 = b2;}
                    else{b2 ^= 0x80; i2 = b2; i2 += 0x80;}
                    lr[2] = (i1<<8) ^ i2;

                    b1 = text[ts+6]; b2 = text[ts+7];
                    if(b1>=0){ i1 = b1;}
                    else{b1 ^= 0x80; i1 = b1; i1 += 0x80;}
                    if(b2>=0){ i2 = b2;}
                    else{b2 ^= 0x80; i2 = b2; i2 += 0x80;}
                    lr[3] = (i1<<8) ^ i2;

                    FL_dec(4,lr , 0, 1, 2, 3);
                    FO_txt(lr, 0, 1, 2, 3, 7, lt, 0, 1, 2);
                    FO_txt(lr, 2, 3, 0, 1, 6, lt, 0, 1, 2);
                    FL_dec(3,lr, 0, 1, 2, 3);
                    FO_txt(lr, 0, 1, 2, 3, 5, lt, 0, 1, 2);
                    FO_txt(lr, 2, 3, 0, 1, 4, lt, 0, 1, 2);
                    FL_dec(2, lr, 0, 1, 2, 3);
                    FO_txt(lr, 0, 1, 2, 3, 3, lt, 0, 1, 2);
                    FO_txt(lr, 2, 3, 0, 1, 2, lt, 0, 1, 2);
                    FL_dec(1,lr, 0, 1, 2, 3);
                    FO_txt(lr, 0, 1, 2, 3, 1, lt, 0, 1, 2);
                    FO_txt(lr, 2, 3, 0, 1, 0, lt, 0, 1, 2);
                    FL_dec(0,lr, 0, 1, 2, 3);

                    text[ts+0] = (byte) (lr[2] >>> 8);
                    text[ts+1] = (byte) (lr[2] & 0xff);
                    text[ts+2] = (byte) (lr[3] >>> 8);
                    text[ts+3] = (byte) (lr[3] & 0xff);
                    text[ts+4] = (byte) (lr[0] >>> 8);
                    text[ts+5] = (byte) (lr[0] & 0xff);
                    text[ts+6] = (byte) (lr[1] >>> 8);
                    text[ts+7] = (byte) (lr[1] & 0xff);

                    ts += 8;
            }
    }
```

```
	}


	//////////////////////////////////////////////////////////////////
	// MistyEC.cpp : コンソール アプリケーションのエントリ ポイントを定義します。
	//
/*
	// 暗号文の HEX 表示用
	char toChar( int c )
	{
		if( c >= 0 && c <= 9 )				// 0～9 ならば
			return char( c + 0x30 );	// ASCII に変換して返す
		else if( c >= 10 && c <= 15 )		// 10～15 ならば
			return char( c + 0x37 );	// A～F の ASCII を返す
		else
			return ' ';
	}
*/
	// メイン

	void MistyEC(String keyfn, String ptfn, String ctfn)
	{
	File fkey, fin, fout;
	int block;
	int i,j,k,len;
	byte[] key = new byte[32];
	byte[] key2 = new byte[64];
	int mode;
	int mesLength =0;
	byte[] bufp;
	int rBlen=0;

	try{
		fkey = new File(keyfn);
		fkey.getParentFile().mkdir();
		FileInputStream inkeyst=null;
		try {
			inkeyst = new FileInputStream(fkey);
			inkeyst.read(key2);
		} catch (IOException e3) {
			// TODO 自動生成された catch ブロック
			e3.printStackTrace();
		}//127


		for(i=0; i<16; i++){
			j = key2[2*i];
			k = key2[2*i+1];
			if(j>=0x30 && j<=0x39) j = j-0x30;
			else{
				if(j>=0x41 && j<=0x46) j = j-0x41+0x0A;
```

```
                }
                if(k>=0x30 && k<=0x39) k = k-0x30;
                else{
                        if(k>=0x41 && k<=0x46) k = k-0x41+0x0A;
                }
                key[i] = (byte) (j*0x10 + k);
        }
        key[16] = 0;//(Byte) null;

        ////////////////////////////////////////////////////////////
        fin = new File(ptfn);
        fin.getParentFile().mkdir();
        FileInputStream inptst=null;
        try {
                inptst = new FileInputStream(fin);
        } catch (FileNotFoundException e5) {
                // TODO 自動生成された catch ブロック
                e5.printStackTrace();
        }

        fout = new File(ctfn);
        fout.getParentFile().mkdir();
        FileOutputStream outctst=null;
        try {
                outctst = new FileOutputStream(fout);
        } catch (FileNotFoundException e5) {
                // TODO 自動生成された catch ブロック
                e5.printStackTrace();
        }

// 暗号文

         int filelen = inptst.available();
         mesLength = filelen + 4;

        if(mesLength <= 1024){
                if((mesLength%8)!=0){
                        block = mesLength/8 + 1;
                }else{
                        block = mesLength/8;
                }
                bufp = new byte[block*8 + 2];

                bufp[0] = (byte) filelen;
                bufp[1] = (byte)(filelen>>>8);
                bufp[2] = (byte)(filelen>>>16);
                bufp[3] = (byte)(filelen>>>24);

        // 暗文
                len = inptst.read(bufp, 4, filelen );
```

```
// 実行
                ts = 0;
                mode = 0;
                misty1( bufp, key,   block,   mode);

                outctst.write(bufp, 0, block*8);
        }
        else
{       // if the file length is more 1024 bytes
        // read the file a block at a time
                if((mesLength%8)!=0){
                        block = mesLength/8 + 1;
                }else{
                        block = mesLength/8;
                }
                bufp = new byte[1024 + 2];

                bufp[0] = (byte) filelen;
                bufp[1] = (byte)(filelen>>>8);
                bufp[2] = (byte)(filelen>>>16);
                bufp[3] = (byte)(filelen>>>24);

                rBlen = block;
                int r =0;
        do //while(rlen > 0 && finst.available()>0)
        {
                // read a block and reduce the remaining byte count
            if(r==0){
                len = inptst.read(bufp, 4, 1024-4);
            }else{
                len = inptst.read(bufp, 0, 1024);
            }

            if(rBlen >= 1024/8){block = 1024/8;}
            if(rBlen < 1024/8){block = rBlen;}

            mode = 0;
            misty1( bufp, key, block, mode);

            outctst.write(bufp, 0, block*8);
            r += 1;
            rBlen -= 1024/8;
        }while(rBlen>0);
}

        if(inkeyst != null)
        {
                try {
                        inkeyst.close();
                } catch (IOException e) {
                        // TODO 自動生成された catch ブロック
```

```
                    e.printStackTrace();
                }
            }

            if(outctst != null)
            {
                try {
                    outctst.close();
                } catch (IOException e) {
                    // TODO 自動生成された catch ブロック
                    e.printStackTrace();
                }
            }

            if(inptst != null)
            {
                try {
                    inptst.close();
                } catch (IOException e) {
                    // TODO 自動生成された catch ブロック
                    e.printStackTrace();
                }
            }

            return;// 0;

}catch (IOException e) {
            // TODO 自動生成された catch ブロック
            e.printStackTrace();
}

}

////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////
// MistyDC.cpp : コンソール アプリケーションのエントリ ポイントを定義します。
//

// 暗号文の HEX 表示用
char toChar( int c )
{
            if( c >= 0 && c <= 9 )                          // 0～9 ならば
                return (char) ( c + 0x30 ); // ASCII に変換して返す
            else if( c >= 10 && c <= 15 )           // 10～15 ならば
                return (char) ( c + 0x37 ); // A～F の ASCII を返す
            else
                return ' ';
}

// メイン
void MistyDC(String keyfn, String ctfn, String ptfn)
```

```
{
        int block;
        int i;
        byte[] key = new byte[32];
        byte[] key2 = new byte[64];
        int j,k;
    int len, rlen;
        int mode;
        int lenp = 0;
        int mesLength =0;
        byte[] bufp;
        int rBlen=0;


        try{
        FileOutputStream foutst = openFileOutput(ptfn,MODE_PRIVATE);
        FileInputStream finst = openFileInput(ctfn);

        File fkey = new File(keyfn);
        fkey.getParentFile().mkdir();
        FileInputStream inkeyst=null;
        try {
                inkeyst = new FileInputStream(fkey);
                inkeyst.read(key2);
        } catch (IOException e3) {
                // TODO 自動生成された catch ブロック
                e3.printStackTrace();
        }//127


        for(i=0; i<16; i++){
                j = key2[2*i];
                k = key2[2*i+1];
                if(j>=0x30 && j<=0x39) j = j-0x30;
                else{
                        if(j>=0x41 && j<=0x46) j = j-0x41+0x0A;
                }
                if(k>=0x30 && k<=0x39) k = k-0x30;
                else{
                        if(k>=0x41 && k<=0x46) k = k-0x41+0x0A;
                }
                key[i] = (byte) (j*0x10 + k);
        }
        key[16] = 0;//(Byte) null;

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

    // 暗号文

         int filelen = finst.available();
         rlen = filelen;
```

```
            mesLength = filelen;

        if(mesLength <= 1024){
                int t =0;
                if((mesLength%8)!=0){
                        block = mesLength/8 + 1;
                }else{
                        block = mesLength/8;
                }
                bufp = new byte[block*8 + 2];

        // 暗文
                len = finst.read(bufp,0,filelen);
                rlen = rlen - len;

    // 復号化実行
                ts =0;
                mode = 1;
                misty1( bufp, key,   block,   mode);

                // 復号文出力
//              pfilelen = *((long*)(pbuf));
                byte[] tmpch4 = new byte[4];
                tmpch4[0] = bufp[0];//(byte) pbuf[0];
                tmpch4[1] = bufp[1];//(byte) (pbuf[0]>>>8);
                tmpch4[2] = bufp[2];//(byte) (pbuf[0]>>>16);
                tmpch4[3] = bufp[3];//(byte) (pbuf[0]>>>24);
                int jj = 0;
                int tmp = 0;
                for (int p = 0; p < tmpch4.length; p++) {
                        tmp = (tmpch4[3-p] & 0xff);
                        if(tmp < 0){
                                tmpch4[3-p] ^= 0x80;
                                tmp = tmpch4[3-p];
                                tmp += 0x80;
                        }
                        jj = (jj << 8) | tmp;
                }
                lenp = jj;

                foutst.write(bufp, 4, lenp);

        }
        else
    {   // if the file length is more 1024 bytes
        // read the file a block at a time
                if((mesLength%8)!=0){
                        block = mesLength/8 + 1;
                }else{
                        block = mesLength/8;
                }
```

```
            bufp = new byte[1024 + 2];

            rBlen = block;
            int r =0;
    do //while(rlen > 0 && finst.available()>0)
    {
        i = 0;
            // read a block and reduce the remaining byte count
        len = finst.read(bufp, 0, 1024);

        if(rBlen >= 1024/8){block = 1024/8;}
        if(rBlen < 1024/8){block = rBlen;}

        mode = 1;
        misty1( bufp, key, block, mode);

        if(r ==0){
            byte[] tmpch4 = new byte[4];
                    tmpch4[0] = bufp[0];//(byte) pbuf[0];
                    tmpch4[1] = bufp[1];//(byte) (pbuf[0]>>>8);
                    tmpch4[2] = bufp[2];//(byte) (pbuf[0]>>>16);
                    tmpch4[3] = bufp[3];//(byte) (pbuf[0]>>>24);
                    int jj = 0;
                    int tmp = 0;
                    for (int p = 0; p < tmpch4.length; p++) {
                            tmp = (tmpch4[3-p] & 0xff);
                            if(tmp < 0){
                                    tmpch4[3-p] ^= 0x80;
                                    tmp = tmpch4[3-p];
                                    tmp += 0x80;
                            }
                            jj = (jj << 8) | tmp;
                    }
                    lenp = jj;
            foutst.write(bufp, 4, 1024-4);
            lenp -= 1024-4;
        }
        else{
            if(rBlen >= 1024/8){
                    foutst.write(bufp, 0, 1024);
                    lenp -= 1024;
            }
            else{
                    foutst.write(bufp, 0, lenp);
            }
        }
        r += 1;
        rBlen -= 1024/8;
    }while(rBlen>0);
}
```

```
                if(inkeyst != null)
                {
                        try {
                                inkeyst.close();
                        } catch (IOException e) {
                                // TODO 自動生成された catch ブロック
                                e.printStackTrace();
                        }
                }

                if(foutst != null)
                {
                        try {
                                foutst.close();
                        } catch (IOException e) {
                                // TODO 自動生成された catch ブロック
                                e.printStackTrace();
                        }
                }

                if(finst != null)
                {
                        try {
                                finst.close();
                        } catch (IOException e) {
                                // TODO 自動生成された catch ブロック
                                e.printStackTrace();
                        }
                }

                return;// 0;

        }catch (IOException e) {
                // TODO 自動生成された catch ブロック
                e.printStackTrace();
        }

        }


}
package yu.com.pcs.jp.sumaho.cg5mail;

import java.util.ArrayList;

import android.app.Activity;
import android.content.Context;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.LinearLayout;
```

```
import android.widget.TextView;

public class MyListAdapter extends BaseAdapter {
   private Context context = null;
   private ArrayList<ListItem> data = null;
   private int resource = 0;

   public MyListAdapter(Context context,
      ArrayList<ListItem> data, int resource) {
      this.context = context;
      this.data = data;
      this.resource = resource;
   }

   public int getCount() {
      return data.size();
   }

   public Object getItem(int position) {
      return data.get(position);
   }

   public long getItemId(int position) {
      return data.get(position).getId();
   }

   public View getView(int position, View convertView, ViewGroup parent) {
      Activity activity = (Activity) context;
      ListItem item = (ListItem) getItem(position);
      LinearLayout v = (LinearLayout) activity.getLayoutInflater()
         .inflate(
         resource, null);
      ((TextView) v.findViewById(R.id.subject)).setText(item.getSubject());
      ((TextView) v.findViewById(R.id.date)).setText(item.getDate());
      ((TextView) v.findViewById(R.id.from)).setText(item.getFrom());
      return v;
   }
}

package yu.com.pcs.jp.sumaho.cg5mail;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.Random;

import yu.com.pcs.jp.sumaho.cg5mail.MailViewActivity.cipherInstance;
import yu.com.pcs.jp.sumaho.cg5mail.MailViewActivity.keyInstance;
```

```
import android.app.Activity;
import android.os.Bundle;
import android.os.Environment;

public class NekoActivity extends Activity{
	String prgn;
	String keyfn;
	String inputfn;
	String outputfn;

	@Override
	protected void onCreate(Bundle savedInstanceState) {
	  super.onCreate(savedInstanceState);
		prgn = "";
	    keyfn = "";
	    inputfn = "";
	    outputfn = "";
	}


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

	// 戻り値
		int NOERROR =    0;                         // エラーなし
		int NOTENOUGHMEMORY =     -1;                          // メモリー不足
		int	ACCESSERROR =         -2;                          // アクセスエラー
		int	MATHERROR    =        -3;                          // 数学的な誤り
		int	KEYLENGTHERROR=   -4;                          // 鍵長不正
		int OTHERERROR         =        -5;                          // その他のエラー

		// 定数
		int MINKEYLENGTH =  32;                          // 最低鍵長（ビット）



	int atoi( byte s[] ) {
		int i, n, sign;
		for( i = 0; s[i] == ' ' ; i++ )   //先頭の空白を読み飛ばす
			;
		sign = ( s[i] == '-' ) ? -1 : 1;      //符号を保存する
		if( s[i] == '-' || s[i] == '+' )      //符号を飛ばす
			i++;
		for( n = 0; i< s.length - 2 ; i++ )   //s[i]が数字のあいだ、n へ
			n = 10 * n + ( s[i] - '0' );
		return sign * n;                          //符号を反映
	}


	// 暗号化    復号化
//	void bmp1(char* St1, char* St2, char* St3);
////////////////////////////////////////////////////////
```

```
// 暗号化
void NekoEC(String keyfn, String ptfn, String ctfn)  // 暗号化
{
        File fkey, fin, fout, fneko;
    int len, rlen, blen4, blen;
        int mode,klen, rc=0;
        int j, k,l, jj;
        int i, mn;
        int nsize, sidesize;
        int fsize;
        byte tmpch4[] = new byte[4];
        byte tmprbuf1[] = new byte[1];
        byte key[] = new byte[64];//32];
        byte[] c_keyb = null;// new byte[66];
        byte c;
        byte FName[] = new byte[256];
        byte[] c_mode = new byte[3];
        byte[] c_klen = new byte[5];
        ////////////////////////////////////

        byte bmpHeader[] = {
                        'B', 'M', /* [ 0] ファイルタイプ */
                        54, 4, 0, 0, /* [ 2] ファイルサイズ  54+4*16*16=1078*/
                        0, 0, 0, 0, /* [ 6] 予約 */
                        54, 0, 0, 0, /* [10] ビットマップデータのシーク位置 */
                        40, 0, 0, 0, /* [14] ここから始まるヘッダの高さ */
                        16, 0, 0, 0, /* [18] ビットマップの幅 */
                        16, 0, 0, 0, /* [22] ビットマップの高さ */
                        0x01, 0, /* [26] プレーン数 */
                        32, 0, /* [28] 1 ピクセルあたりのビット数 （課題が 4 バイト指定されていたので 32bit に変更） */
                        0, 0, 0, 0, /* [30] 圧縮タイプ */
                        0, 1, 0, 0, /* [34] ビットマップデータの長さ 16*16=256*/
                        0, 0, 0, 0, /* [38] 水平解像度(px/m) */
                        0, 0, 0, 0, /* [42] 垂直解像度(px/m) */
                        0, 0, 0, 0, /* [46] カラーインデックス数 */
                        0, 0, 0, 0, /* [50] 重要なカラーインデックス数 */
        };

        byte bmpHeader2[] = {
                        'B', 'M', /* [ 0] ファイルタイプ */
                        54, 4, 0, 0, /* [ 2] ファイルサイズ  54+4*16*16=1078*/
                        0, 0, 0, 0, /* [ 6] 予約 */
                        54, 0, 0, 0, /* [10] ビットマップデータのシーク位置 */
                        40, 0, 0, 0, /* [14] ここから始まるヘッダの高さ */
                        16, 0, 0, 0, /* [18] ビットマップの幅 */
                        16, 0, 0, 0, /* [22] ビットマップの高さ */
                        0x01, 0, /* [26] プレーン数 */
                        32, 0, /* [28] 1 ピクセルあたりのビット数 （課題が 4 バイト指定されていたので 32bit に変更） */
```

```
                    0, 0, 0, 0, /* [30] 圧縮タイプ */
                    0, 1, 0, 0, /* [34] ビットマップデータの長さ 16*16=256*/
                    0, 0, 0, 0, /* [38] 水平解像度(px/m) */
                    0, 0, 0, 0, /* [42] 垂直解像度(px/m) */
                    0, 0, 0, 0, /* [46] カラーインデックス数 */
                    0, 0, 0, 0, /* [50] 重要なカラーインデックス数 */
};


try{
String kf1 = Environment.getExternalStorageDirectory() + "/" + "nyanya2.bmp";

fneko = new File(kf1);
fneko.getParentFile().mkdir();
FileInputStream innekost = null;
innekost = new FileInputStream(fneko);
innekost.read(bmpHeader2);

tmpch4[0] = bmpHeader2[18];//(byte) pbuf[0];
tmpch4[1] = bmpHeader2[19];//(byte) (pbuf[0]>>>8);
tmpch4[2] = bmpHeader2[20];//(byte) (pbuf[0]>>>16);
tmpch4[3] = bmpHeader2[21];//(byte) (pbuf[0]>>>24);
jj = 0;
int tmp = 0;
for (int p = 0; p < tmpch4.length; p++) {
        tmp = (tmpch4[3-p] & 0xff);
        if(tmp < 0){
                tmpch4[3-p] ^= 0x80;
                tmp = tmpch4[3-p];
                tmp += 0x80;
        }
        jj = (jj << 8) | tmp;
}
int nhsize = jj;

tmpch4[0] = bmpHeader2[22];//(byte) pbuf[0];
tmpch4[1] = bmpHeader2[23];//(byte) (pbuf[0]>>>8);
tmpch4[2] = bmpHeader2[24];//(byte) (pbuf[0]>>>16);
tmpch4[3] = bmpHeader2[25];//(byte) (pbuf[0]>>>24);
jj = 0;
tmp = 0;
for (int p = 0; p < tmpch4.length; p++) {
        tmp = (tmpch4[3-p] & 0xff);
        if(tmp < 0){
                tmpch4[3-p] ^= 0x80;
                tmp = tmpch4[3-p];
                tmp += 0x80;
        }
        jj = (jj << 8) | tmp;
}
int nvsize = jj;
```

```
tmpch4[0] = bmpHeader2[2];//(byte) pbuf[0];
tmpch4[1] = bmpHeader2[3];//(byte) (pbuf[0]>>>8);
tmpch4[2] = bmpHeader2[4];//(byte) (pbuf[0]>>>16);
tmpch4[3] = bmpHeader2[5];//(byte) (pbuf[0]>>>24);
jj = 0;
tmp = 0;
for (int p = 0; p < tmpch4.length; p++) {
        tmp = (tmpch4[3-p] & 0xff);
        if(tmp < 0){
                tmpch4[3-p] ^= 0x80;
                tmp = tmpch4[3-p];
                tmp += 0x80;
        }
        jj = (jj << 8) | tmp;
}
int nfsize = jj;

fkey = new File(keyfn);
fkey.getParentFile().mkdir();
FileInputStream inkeyst=null;
try {
        inkeyst = new FileInputStream(fkey);
        inkeyst.read( c_mode);
        inkeyst.read( c_klen);
        klen = atoi(c_klen);
        c_keyb = new byte[klen/4+2];
        inkeyst.read( c_keyb );
} catch (IOException e5) {
        // TODO 自動生成された catch ブロック
        e5.printStackTrace();
}

mode = atoi(c_mode);
klen = atoi(c_klen);

if(klen<56 || 256<klen){
        //                      printf("Wrong key size. ¥n");
        return ;
}

for( i=0; i<klen/4;i++){
        key[i] = c_keyb[i];
}


mn = 8;
if(klen == 128){ mn = 16; }
if(klen == 192){ mn = 24; }
if(klen == 256){ mn = 32; }
```

```
fin = new File(ptfn);
fin.getParentFile().mkdir();
FileInputStream inptst=null;
try {
	inptst = new FileInputStream(fin);
} catch (FileNotFoundException e5) {
	// TODO 自動生成された catch ブロック
	e5.printStackTrace();
}

fout = new File(ctfn);
fout.getParentFile().mkdir();
FileOutputStream outctst=null;
try {
	outctst = new FileOutputStream(fout);
} catch (FileNotFoundException e5) {
	// TODO 自動生成された catch ブロック
	e5.printStackTrace();
}

Random rand = new Random();   //in constracter

//		FName = ptfn);// = srcfile.GetFileName();
		nsize = ptfn.length();//strlen(FName); //.GetLength();// length of file name.
		// 平文
//		fseek(srcfile, 0, SEEK_END);
		int filelen;

		filelen = inptst.available();
//		fseek(srcfile,0,0);
		fsize = filelen;	//fsize = srcfile.GetLength();// as char 8 bit
		sidesize = ((4+1+(fsize/2)+1+(nsize/2))/nhsize) + 1;
		//sidesize = 1 + (int)sqrt((double)(4+1+(fsize/2)+1+(nsize/2)));// as short int 16 bit


		long f_size = 54+4*sidesize*(sidesize + nvsize);
		bmpHeader[2] = (byte)(f_size);
		bmpHeader[3] = (byte)(f_size/0x100);
		bmpHeader[4] = (byte)(f_size/0x10000);
		bmpHeader[5] = (byte)(f_size/0x1000000);

		bmpHeader[18] = (byte)(nhsize);
		bmpHeader[19] = (byte)(nhsize/0x100);
		bmpHeader[20] = (byte)(nhsize/0x10000);
		bmpHeader[21] = (byte)(nhsize/0x1000000);

		bmpHeader[22] = (byte)(sidesize + nvsize);
		bmpHeader[23] = (byte)((sidesize + nvsize)/0x100);
		bmpHeader[24] = (byte)((sidesize + nvsize)/0x10000);
		bmpHeader[25] = (byte)((sidesize + nvsize)/0x1000000);
```

```
f_size = sidesize*sidesize;
bmpHeader[34] = (byte)(sidesize*(sidesize + nvsize));
bmpHeader[35] = (byte)(sidesize*(sidesize + nvsize)/0x100);
bmpHeader[36] = (byte)(sidesize*(sidesize + nvsize)/0x10000);
bmpHeader[37] = (byte)(sidesize*(sidesize + nvsize)/0x1000000);

outctst.write(bmpHeader,0,54);


j = rand.nextInt(65535); // 16 bits
k = nsize & 0x0000ffff;
jj = (j<<16) | (j^k);
tmpch4[0] = (byte)(jj);
tmpch4[1] = (byte)(jj>>>8);///0x100);
tmpch4[2] = (byte)(jj>>>16);///0x10000);
tmpch4[3] = (byte)(jj>>>24);///0x1000000);
outctst.write(tmpch4,0,4);

j = rand.nextInt(65535);// 16 bits
k = nsize & 0xffff0000;
jj = (j<<16) | (j^(k>>>16));
tmpch4[0] = (byte)(jj);
tmpch4[1] = (byte)(jj>>>8);///0x100);
tmpch4[2] = (byte)(jj>>>16);///0x10000);
tmpch4[3] = (byte)(jj>>>24);//0x1000000);
outctst.write(tmpch4,0,4);

j = rand.nextInt(65535);// 16 bits
k = (int)(fsize & 0x0000ffff);
jj = (j<<16) | (j^k);
tmpch4[0] = (byte)(jj);
tmpch4[1] = (byte)(jj>>>8);///0x100);
tmpch4[2] = (byte)(jj>>>16);///0x10000);
tmpch4[3] = (byte)(jj>>>24);//0x1000000);
outctst.write(tmpch4,0,4);

j = rand.nextInt(65535); // 16 bits
k = (int)(fsize & 0xffff0000);
jj = (j<<16) | (j^(k>>>16));
tmpch4[0] = (byte)(jj);
tmpch4[1] = (byte)(jj>>>8);///0x100);
tmpch4[2] = (byte)(jj>>>16);///0x10000);
tmpch4[3] = (byte)(jj>>>24);///0x1000000);
outctst.write(tmpch4);

for(i=0; i<nsize/2 ; i++){
        j = rand.nextInt(65535); // 16 bits
        k = FName[2*i];
        l = FName[2*i+1];
        jj = (j<<16) | (j^((k<<8) | l));
        tmpch4[0] = (byte)(jj);
```

```
            tmpch4[1] = (byte)(jj>>>8);///0x100);
            tmpch4[2] = (byte)(jj>>>16);///0x10000);
            tmpch4[3] = (byte)(jj>>>24);//0x1000000);
            outctst.write(tmpch4,0,4);

}
if(nsize%2 == 1){
            j = rand.nextInt(65535); // 16 bits
            k = FName[nsize-1];
            jj = (j<<16) | (j^((k<<8) | 0));
            tmpch4[0] = (byte)(jj);
            tmpch4[1] = (byte)(jj>>>8);///0x100);
            tmpch4[2] = (byte)(jj>>>16);///0x10000);
            tmpch4[3] = (byte)(jj>>>24);//0x1000000);
            outctst.write(tmpch4,0,4);
}
if(nsize%2 == 0){
            j = rand.nextInt(65535); // 16 bits
            k = 0;
            jj = (j<<16) | (j^((k<<8) | 0));
            tmpch4[0] = (byte)(jj);
            tmpch4[1] = (byte)(jj>>>8);///0x100);
            tmpch4[2] = (byte)(jj>>>16);///0x10000);
            tmpch4[3] = (byte)(jj>>>24);//0x1000000);
            outctst.write(tmpch4,0,4);
}

for(i=4+1+nsize/2; i<sidesize*nhsize ; i++){
            j = rand.nextInt(65535); // 16 bits

            if(1 == inptst.read(tmprbuf1)){
                        k = tmprbuf1[0];
                        if(k<0){
                                    tmprbuf1[0] ^= 0x80;
                                    k = tmprbuf1[0];
                                    k += 0x80;
                        }
                        k ^= key[(i-(4+1+nsize/2))%mn];
            }
            else{k = 0;}

            if(1 == inptst.read(tmprbuf1)){
                        l = tmprbuf1[0];
                        if(l<0){
                                    tmprbuf1[0] ^= 0x80;
                                    l = tmprbuf1[0];
                                    l += 0x80;
                        }
                        l ^= key[(i-(4+1+nsize/2))%mn];
                        jj = (j<<16) | (j^((k<<8) | l));
            }
```

```
                    else{jj = (j<<16) | (j^((k<<8) | 0));}

                    tmpch4[0] = (byte)(jj);
                    tmpch4[1] = (byte)(jj>>>8);//0x100);
                    tmpch4[2] = (byte)(jj>>>16);///0x10000);
                    tmpch4[3] = (byte)(jj>>>24);//0x1000000);
                    outctst.write(tmpch4,0,4);

                }

                tmpch4[3] = 0;
                for(i=0; i<nhsize*nvsize; i++){//遅い！！
                    innekost.read(tmpch4,0,3);
                    outctst.write(tmpch4,0,4);
                }

                outctst.flush();

                if (innekost != null)
                    innekost.close();
                if (outctst != null)
                    outctst.close();
                if (inptst != null)
                    inptst.close();

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


        }


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


// 復号化
        void NekoDC(String keyfn, String ctfn, String ptfn)   // 復号化
        {
                int i, jj, tmp;
                byte[] c_mode = new byte[3];
                byte[] c_klen = new byte[5];
                byte[] c_keyb = null;//new byte[64+2];
                byte tmpch4[] = new byte[4];
                int j,k;
                byte tmprbuf1[] = new byte[1];
                byte tmprbuf2[] = new byte[2];
            int len, rlen, blen4, pfilelen;
                int mode,klen,blen,rc=0;
                byte key[] = new byte[64];//32];
                int nsize, fsize, sidesize;
```

```
String FName;//[256];

int   mn;
int q;


byte bmpHeader[] = {
                'B', 'M', /* [ 0] ファイルタイプ */
                54, 4, 0, 0, /* [ 2] ファイルサイズ  54+4*16*16=1078*/
                0, 0, 0, 0, /* [ 6] 予約 */
                54, 0, 0, 0, /* [10] ビットマップデータのシーク位置 */
                40, 0, 0, 0, /* [14] ここから始まるヘッダの高さ */
                16, 0, 0, 0, /* [18] ビットマップの幅 */
                16, 0, 0, 0, /* [22] ビットマップの高さ */
                0x01, 0, /* [26] プレーン数 */
                32, 0, /* [28] 1 ピクセルあたりのビット数 （課題が 4 バイト指定されていたので 32bit に変更） */
                0, 0, 0, 0, /* [30] 圧縮タイプ */
                0, 1, 0, 0, /* [34] ビットマップデータの長さ 16*16=256*/
                0, 0, 0, 0, /* [38] 水平解像度(px/m) */
                0, 0, 0, 0, /* [42] 垂直解像度(px/m) */
                0, 0, 0, 0, /* [46] カラーインデックス数 */
                0, 0, 0, 0, /* [50] 重要なカラーインデックス数 */
};


try{
        FileOutputStream writer2 = openFileOutput(ptfn,MODE_PRIVATE);
        FileInputStream in2 = openFileInput(ctfn);

        File fkey = new File(keyfn);
        fkey.getParentFile().mkdir();
        FileInputStream inkeyst=null;
        try {
                inkeyst = new FileInputStream(fkey);
                inkeyst.read(c_mode);
                inkeyst.read(c_klen);
                klen = atoi(c_klen);
                c_keyb = new byte[klen/4+2];
                inkeyst.read(c_keyb);
        } catch (IOException e3) {
                // TODO 自動生成された catch ブロック
                e3.printStackTrace();
        }//127

        mode = atoi(c_mode);
        klen = atoi(c_klen);

        if(klen<56 || 256<klen){
                //                      printf("Wrong key size. ¥n");
                return ;
```

```
                    }

                    for( i=0; i<klen/4;i++){
                              key[i] = c_keyb[i];
                    }

                    mn = 8;
                    if(klen == 128){ mn = 16; }
                    if(klen == 192){ mn = 24; }
                    if(klen == 256){ mn = 32; }

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

                    if(in2.available() > 0){
                              in2.read(bmpHeader);
                    }

                    if(in2.available() > 0){
                              in2.read(tmpch4);
                    }
                    jj = 0;
                    for (int p = 0; p < tmpch4.length; p++) {
                              tmp = (int)(tmpch4[3-p] & 0xff);
                              jj = (jj << 8) | tmp;
                    }
                    j = jj^(jj>>>16);
                    nsize = j & 0x0000ffff;

                    if(in2.available() > 0){
                              in2.read(tmpch4);
                    }
                    jj = 0;
                    for (int p = 0; p < tmpch4.length; p++) {
                              tmp = (int)(tmpch4[3-p] & 0xff);
                              jj = (jj << 8) | tmp;
                    }
                    j = jj^(jj>>>16);
                    k = j & 0x0000ffff;
                    nsize = nsize + (k<<16);

                    if(in2.available() > 0){
                              in2.read(tmpch4);
                    }
                    jj = 0;
                    for (int p = 0; p < tmpch4.length; p++) {
                              jj = (jj << 8) | (tmpch4[3-p] & 0xff);
                    }
                    j = jj^(jj>>>16);
                    fsize = j & 0x0000ffff;

                    if(in2.available() > 0){
```

```
            in2.read(tmpch4);
}
jj = 0;
for (int p = 0; p < tmpch4.length; p++) {
            jj = (jj << 8) | (tmpch4[3-p] & 0xff);
}
j = jj^(jj>>>16);
k = j & 0x0000ffff;
fsize = fsize + (k<<16);

byte pfName[] = new byte[nsize+8];
if(pfName == null){
            return;
}

for(i=0; i<nsize/2 ; i++){
            if(in2.available() > 0){
                        in2.read(tmpch4);
            }
            jj = 0;
            for (int p = 0; p < tmpch4.length; p++) {
                        jj = (jj << 8) | (tmpch4[3-p] & 0xff);
            }
            j = jj^(jj>>16);
            k = j & 0x0000ffff;
            pfName[2*i] = (byte)(k>>8);
            pfName[2*i+1] = (byte)(k&0x000000ff);
}
for(i=nsize/2; i<(nsize+5)/2 ; i++){
            pfName[2*i] = 0;//(Byte) null;
            pfName[2*i+1] = 0;//(Byte) null;
}
if(nsize%2 == 0){
            if(in2.available() > 0){
                        in2.read(tmpch4);
            }
}
if(nsize%2 == 1){
            if(in2.available() > 0){
                        in2.read(tmpch4);
            }
            jj = 0;
            for (int p = 0; p < tmpch4.length; p++) {
                        jj = (jj << 8) | (tmpch4[3-p] & 0xff);
            }
            j = jj^(jj>>16);
            k = j & 0x0000ffff;
            pfName[nsize-1] = (byte)(k>>8);
}
```

```
                    for(i=0 ; i<fsize ; i+=2){
                        q = fsize - i - 2;
                        if(q >= 0){
                            if(4 == in2.read(tmpch4)){
                                jj = 0;
                                for (int p = 0; p < tmpch4.length; p++) {
                                    jj = (jj << 8) | (tmpch4[3-p] & 0xff);
                                }
                                j = jj^(jj>>16);
                                k = j & 0x0000ffff;
                                tmprbuf2[0] = (byte)(k>>8);
                                tmprbuf2[0] ^= (byte)key[(i/2)%mn];
                                tmprbuf2[1] = (byte)(k&0x000000ff);
                                tmprbuf2[1] ^= (byte)key[(i/2)%mn];
                                writer2.write(tmprbuf2);
                            }
                        }
                        if(q < 0){ // q==-1
                            if(4 == in2.read(tmpch4)){
                                jj = 0;
                                for (int p = 0; p < tmpch4.length; p++) {
                                    jj = (jj << 8) | (tmpch4[3-p] & 0xff);
                                }
                                j = jj^(jj>>16);
                                k = j & 0x0000ffff;
                                tmprbuf1[0] = (byte)(k>>8);
                                tmprbuf1[0] ^= (byte)key[(i/2)%mn];
                                writer2.write(tmprbuf1);
                            }
                        }
                    }

                    writer2.flush();

                    if (writer2 != null)
                        writer2.close();
                    if (in2 != null)
                        in2.close();

                } catch (FileNotFoundException e) {
                    e.printStackTrace();
                }catch (IOException e) {
                    System.out.println("添付ファイルの保存に失敗しました。" + e);
                } finally {

                }
            }

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

```
}
package yu.com.pcs.jp.sumaho.cg5mail;

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

import yu.com.pcs.jp.sumaho.cg5mail.MailViewActivity.cipherInstance;
import yu.com.pcs.jp.sumaho.cg5mail.MailViewActivity.keyInstance;

import android.app.Activity;
import android.os.Bundle;

public class SerpentActivity extends Activity{
	String prgn;
	String keyfn;
	String inputfn;
	String outputfn;

	@Override
	protected void onCreate(Bundle savedInstanceState) {
	  super.onCreate(savedInstanceState);
		prgn = "";
	    keyfn = "";
	    inputfn = "";
	    outputfn = "";
	}


	  /*
		#define BLOCK_SIZE 128
		#define DWORD unsigned long
	*/
		////////////////////////////////////

//		#define	DIR_ENCRYPT	0	/*	Are we encrpyting?	*/
//		#define	DIR_DECRYPT	1	/*	Are we decrpyting?	*/
//		#define	MODE_ECB	1	/*	Are we ciphering in ECB mode?	*/
//		#define	MODE_CBC	2	/*	Are we ciphering in CBC mode?	*/
//		#define	MODE_CFB1	3	/*	Are we ciphering in 1-bit CFB mode? */
//		#define	TRUE		1
//		#define	FALSE		0

		/*	Error Codes - CHANGE POSSIBLE: inclusion of additional error codes	*/
//		#define	BAD_KEY_DIR	-1	/*	Key direction is invalid, e;g;,
//						unknown value */
//		#define	BAD_KEY_MAT	-2	/*	Key material not of correct
//						length */
```

```
//                  #define       BAD_KEY_INSTANCE    -3  /*  Key passed is not valid  */
//                  #define       BAD_CIPHER_MODE     -4  /*  Params struct passed to
//                                                            cipherInit invalid */
//                  #define       BAD_CIPHER_STATE    -5  /*  Cipher in wrong state (e.g., not
//                                                            initialized) */

                    /*  CHANGE POSSIBLE:  inclusion of algorithm specific defines  */
//                  #define       MAX_KEY_SIZE      64  /* # of ASCII char's needed to
//                                                            represent a key */
//                  #define       MAX_IV_SIZE           32  /* # of ASCII char's needed to
//                                                            represent an IV  */

//                  typedef     unsigned char     BYTE;

            /*  The structure for key information */
            class keyInstance {
                    public int  direction;      /*  Key used for encrypting or decrypting? */
                    public int   keyLen;        /*  Length of the key  */
                    public int[]  keyMaterial = new int[64+1];  /*  Raw key data in ASCII, e.g.,
                                                            what the user types or KAT values)*/
                    /*  The following parameters are algorithm dependent, replace or
                            add as necessary  */
                    public int[] key = new int[8];              /* The key in binary */
                    public int[][] subkeys = new int[33][4];    /* Serpent subkeys */
                    }

            /*  The structure for cipher information */
            class cipherInstance {
                    public int  mode;           /* MODE_ECB, MODE_CBC, or MODE_CFB1 */
                    public int[] IVi = new int[32/4];   /* A possible Initialization Vector for
                                                            ciphering */
                    public byte[] IVb = new byte[32];
                    /*  Add any algorithm specific parameters needed here  */
                    public int   blockSize;     /* Sample: Handles non-128 bit block sizes
                                                            (if available) */
                    }

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

            int ob=0, oip=0, obmode=0;
            int[] ox = new int[128/32];
            int[] pbuf = new int[2048/4];
            int[] cbuf = new int[2048/4];
            byte[] cbufb = new byte[2048];
            byte[] pbufb = new byte[2048];
            int[] x = new int[4];


            /* S0:   3  8 15  1 10  6  5 11 14 13  4  2  7  0  9 12 */
            /* depth = 5,7,4,2, Total gates=18 */
            void RND00(int[] p, int a, int b, int c, int d, int[] k, int kw, int kx, int ky, int kz)
```

```
        { int t02, t03, t05, t06, t07, t08, t09, t11, t12, t13, t14, t15, t17, t01;
        int w,x,y,z;
        t01 = p[b]     ^ p[c]   ;
        t02 = p[a]     | p[d]   ;
        t03 = p[a]     ^ p[b]   ;
        z     = t02 ^ t01;
        t05 = p[c]     | z    ;
        t06 = p[a]     ^ p[d]   ;
        t07 = p[b]     | p[c]   ;
        t08 = p[d]     & t05;
        t09 = t03 & t07;
        y     = t09 ^ t08;
        t11 = t09 & y    ;
        t12 = p[c]     ^ p[d]   ;
        t13 = t07 ^ t11;
        t14 = p[b]     & t06;
        t15 = t06 ^ t13;
        w     =        ~ t15;
        t17 = w      ^ t14;
        x     = t12 ^ t17;
        k[kw]=w;
        k[kx]=x;
        k[ky]=y;
        k[kz]=z;
        }

/* InvS0:   13   3 11   0 10   6   5 12   1 14   4   7 15   9   8   2 */
/* depth = 8,4,3,6, Total gates=19 */
void InvRND00(int[] p, int a, int b, int c, int d, int[] k, int kw, int kx, int ky, int kz)
        { int t02, t03, t04, t05, t06, t08, t09, t10, t12, t13, t14, t15, t17, t18, t01;
        int w,x,y,z;
        t01 = p[c]     ^ p[d]   ;
        t02 = p[a]     | p[b]   ;
        t03 = p[b]     | p[c]   ;
        t04 = p[c]     & t01;
        t05 = t02 ^ t01;
        t06 = p[a]     | t04;
        y     =        ~ t05;
        t08 = p[b]     ^ p[d]   ;
        t09 = t03 & t08;
        t10 = p[d]     | y    ;
        x     = t09 ^ t06;
        t12 = p[a]     | t05;
        t13 = x      ^ t12;
        t14 = t03 ^ t10;
        t15 = p[a]     ^ p[c]   ;
        z     = t14 ^ t13;
        t17 = t05 & t13;
        t18 = t14 | t17;
        w     = t15 ^ t18;
        k[kw]=w;
```

```
        k[kx]=x;
        k[ky]=y;
        k[kz]=z;
        }

/* S1:  15 12  2  7  9  0  5 10  1 11 14  8  6 13  3  4 */
/* depth = 10,7,3,5, Total gates=18 */
void RND01(int[] p, int a, int b, int c, int d, int[] k, int kw, int kx, int ky, int kz)
        { int t02, t03, t04, t05, t06, t07, t08, t10, t11, t12, t13, t16, t17, t01;
        int w,x,y,z;
        t01 = p[a]    | p[d]   ;
        t02 = p[c]    ^ p[d]   ;
        t03 =       ~ p[b]   ;
        t04 = p[a]    ^ p[c]   ;
        t05 = p[a]    | t03;
        t06 = p[d]    & t04;
        t07 = t01 & t02;
        t08 = p[b]    | t06;
        y    = t02 ^ t05;
        t10 = t07 ^ t08;
        t11 = t01 ^ t10;
        t12 = y    ^ t11;
        t13 = p[b]    & p[d]   ;
        z    =       ~ t10;
        x    = t13 ^ t12;
        t16 = t10 | x   ;
        t17 = t05 & t16;
        w    = p[c]    ^ t17;
        k[kw]=w;
        k[kx]=x;
        k[ky]=y;
        k[kz]=z;
        }

/* InvS1:   5  8  2 14 15  6 12  3 11  4  7  9  1 13 10  0 */
/* depth = 7,4,5,3, Total gates=18 */
void InvRND01(int[] p, int a, int b, int c, int d, int[] k, int kw, int kx, int ky, int kz)
        { int t02, t03, t04, t05, t06, t07, t08, t09, t10, t11, t14, t15, t17, t01;
        int w,x,y,z;
        t01 = p[a]    ^ p[b]   ;
        t02 = p[b]    | p[d]   ;
        t03 = p[a]    & p[c]   ;
        t04 = p[c]    ^ t02;
        t05 = p[a]    | t04;
        t06 = t01 & t05;
        t07 = p[d]    | t03;
        t08 = p[b]    ^ t06;
        t09 = t07 ^ t06;
        t10 = t04 | t03;
        t11 = p[d]    & t08;
        y    =       ~ t09;
```

```
        x    = t10 ^ t11;
        t14 = p[a]    | y   ;
        t15 = t06 ^ x   ;
        z    = t01 ^ t04;
        t17 = p[c]    ^ t15;
        w    = t14 ^ t17;
        k[kw]=w;
        k[kx]=x;
        k[ky]=y;
        k[kz]=z;
        }

/* S2:    8   6   7   9   3 12 10 15 13   1 14   4   0 11   5   2 */
/* depth = 3,8,11,7, Total gates=16 */
void RND02(int[] p, int a, int b, int c, int d, int[] k, int kw, int kx, int ky, int kz)
        { int t02, t03, t05, t06, t07, t08, t09, t10, t12, t13, t14, t01;
        int w,x,y,z;
        t01 = p[a]    | p[c]   ;
        t02 = p[a]    ^ p[b]   ;
        t03 = p[d]    ^ t01;
        w    = t02 ^ t03;
        t05 = p[c]    ^ w   ;
        t06 = p[b]    ^ t05;
        t07 = p[b]    | t05;
        t08 = t01 & t06;
        t09 = t03 ^ t07;
        t10 = t02 | t09;
        x    = t10 ^ t08;
        t12 = p[a]    | p[d]   ;
        t13 = t09 ^ x   ;
        t14 = p[b]    ^ t13;
        z    =       ~ t09;
        y    = t12 ^ t14;
        k[kw]=w;
        k[kx]=x;
        k[ky]=y;
        k[kz]=z;
        }

/* InvS2:   12   9 15   4 11 14   1   2   0   3   6 13   5   8 10   7 */
/* depth = 3,6,8,3, Total gates=18 */
void InvRND02(int[] p, int a, int b, int c, int d, int[] k, int kw, int kx, int ky, int kz)
        { int t02, t03, t04, t06, t07, t08, t09, t10, t11, t12, t15, t16, t17, t01;
        int w,x,y,z;
        t01 = p[a]    ^ p[d]   ;
        t02 = p[c]    ^ p[d]   ;
        t03 = p[a]    & p[c]   ;
        t04 = p[b]    | t02;
        w    = t01 ^ t04;
        t06 = p[a]    | p[c]   ;
        t07 = p[d]    | w   ;
```

```
        t08 =       ~ p[d]   ;
        t09 = p[b]     & t06;
        t10 = t08 | t03;
        t11 = p[b]     & t07;
        t12 = t06 & t02;
        z     = t09 ^ t10;
        x     = t12 ^ t11;
        t15 = p[c]     & z   ;
        t16 = w      ^ x   ;
        t17 = t10 ^ t15;
        y     = t16 ^ t17;
        k[kw]=w;
        k[kx]=x;
        k[ky]=y;
        k[kz]=z;
        }

/* S3:    0 15 11   8 12   9   6   3 13   1   2   4 10   7   5 14 */
/* depth = 8,3,5,5, Total gates=18 */
void RND03(int[] p, int a, int b, int c, int d, int[] k, int kw, int kx, int ky, int kz)
        { int t02, t03, t04, t05, t06, t07, t08, t09, t10, t11, t13, t14, t15, t01;
        int w,x,y,z;
        t01 = p[a]     ^ p[c]   ;
        t02 = p[a]     | p[d]   ;
        t03 = p[a]     & p[d]   ;
        t04 = t01 & t02;
        t05 = p[b]     | t03;
        t06 = p[a]     & p[b]   ;
        t07 = p[d]     ^ t04;
        t08 = p[c]     | t06;
        t09 = p[b]     ^ t07;
        t10 = p[d]     & t05;
        t11 = t02 ^ t10;
        z     = t08 ^ t09;
        t13 = p[d]     | z   ;
        t14 = p[a]     | t07;
        t15 = p[b]     & t13;
        y     = t08 ^ t11;
        w     = t14 ^ t15;
        x     = t05 ^ t04;
        k[kw]=w;
        k[kx]=x;
        k[ky]=y;
        k[kz]=z;
        }

/* InvS3:    0   9 10   7 11 14   6 13   3   5 12   2   4   8 15   1 */

/* depth = 3,6,4,4, Total gates=17 */
void InvRND03(int[] p, int a, int b, int c, int d, int[] k, int kw, int kx, int ky, int kz)
        { int t02, t03, t04, t05, t06, t07, t09, t11, t12, t13, t14, t16, t01;
```

```
        int w,x,y,z;
        t01 = p[c]    | p[d]   ;
        t02 = p[a]    | p[d]   ;
        t03 = p[c]    ^ t02;
        t04 = p[b]    ^ t02;
        t05 = p[a]    ^ p[d]   ;
        t06 = t04 & t03;
        t07 = p[b]    & t01;
        y    = t05 ^ t06;
        t09 = p[a]    ^ t03;
        w    = t07 ^ t03;
        t11 = w     | t05;
        t12 = t09 & t11;
        t13 = p[a]    & y   ;
        t14 = t01 ^ t05;
        x    = p[b]    ^ t12;
        t16 = p[b]    | t13;
        z    = t14 ^ t16;
        k[kw]=w;
        k[kx]=x;
        k[ky]=y;
        k[kz]=z;
        }

/* S4:    1 15   8   3 12   0 11   6   2   5   4 10   9 14   7 13 */
/* depth = 6,7,5,3, Total gates=19 */
void RND04(int[] p, int a, int b, int c, int d, int[] k, int kw, int kx, int ky, int kz)
        { int t02, t03, t04, t05, t06, t08, t09, t10, t11, t12, t13, t14, t15, t16, t01;
        int w,x,y,z;
        t01 = p[a]    | p[b]   ;
        t02 = p[b]    | p[c]   ;
        t03 = p[a]    ^ t02;
        t04 = p[b]    ^ p[d]   ;
        t05 = p[d]    | t03;
        t06 = p[d]    & t01;
        z    = t03 ^ t06;
        t08 = z    & t04;
        t09 = t04 & t05;
        t10 = p[c]    ^ t06;
        t11 = p[b]    & p[c]   ;
        t12 = t04 ^ t08;
        t13 = t11 | t03;
        t14 = t10 ^ t09;
        t15 = p[a]    & t05;
        t16 = t11 | t12;
        y    = t13 ^ t08;
        x    = t15 ^ t16;
        w    =       ~ t14;
        k[kw]=w;
        k[kx]=x;
        k[ky]=y;
```

```
        k[kz]=z;
        }

/* InvS4:    5   0   8   3 10   9   7 14   2 12 11   6   4 15 13   1 */

/* depth = 6,4,7,3, Total gates=17 */
void InvRND04(int[] p, int a, int b, int c, int d, int[] k, int kw, int kx, int ky, int kz)
        { int t02, t03, t04, t05, t06, t07, t09, t10, t11, t12, t13, t15, t01;
        int w,x,y,z;
        t01 = p[b]    | p[d]   ;
        t02 = p[c]    | p[d]   ;
        t03 = p[a]    & t01;
        t04 = p[b]    ^ t02;
        t05 = p[c]    ^ p[d]   ;
        t06 =        ~ t03;
        t07 = p[a]    & t04;
        x    = t05 ^ t07;
        t09 = x     | t06;
        t10 = p[a]    ^ t07;
        t11 = t01 ^ t09;
        t12 = p[d]    ^ t04;
        t13 = p[c]    | t10;
        z    = t03 ^ t12;
        t15 = p[a]    ^ t04;
        y    = t11 ^ t13;
        w    = t15 ^ t09;
        k[kw]=w;
        k[kx]=x;
        k[ky]=y;
        k[kz]=z;
        }

/* S5:   15   5   2 11   4 10   9 12   0   3 14   8 13   6   7   1 */
/* depth = 4,6,8,6, Total gates=17 */
void RND05(int[] p, int a, int b, int c, int d, int[] k, int kw, int kx, int ky, int kz)
        { int t02, t03, t04, t05, t07, t08, t09, t10, t11, t12, t13, t14, t01;
        int w,x,y,z;
        t01 = p[b]    ^ p[d]   ;
        t02 = p[b]    | p[d]   ;
        t03 = p[a]    & t01;
        t04 = p[c]    ^ t02;
        t05 = t03 ^ t04;
        w    =        ~ t05;
        t07 = p[a]    ^ t01;
        t08 = p[d]    | w   ;
        t09 = p[b]    | t05;
        t10 = p[d]    ^ t08;
        t11 = p[b]    | t07;
        t12 = t03 | w   ;
        t13 = t07 | t10;
        t14 = t01 ^ t11;
```

```
        y    = t09 ^ t13;
        x    = t07 ^ t08;
        z    = t12 ^ t14;
        k[kw]=w;
        k[kx]=x;
        k[ky]=y;
        k[kz]=z;
        }

/* InvS5:    8 15   2   9   4   1 13 14 11   6   5   3   7 12 10   0 */
/* depth = 4,6,9,7, Total gates=17 */
void InvRND05(int[] p, int a, int b, int c, int d, int[] k, int kw, int kx, int ky, int kz)
        { int t02, t03, t04, t05, t07, t08, t09, t10, t12, t13, t15, t16, t01;
        int w,x,y,z;
        t01 = p[a]    & p[d]   ;
        t02 = p[c]    ^ t01;
        t03 = p[a]    ^ p[d]   ;
        t04 = p[b]    & t02;
        t05 = p[a]    & p[c]   ;
        w    = t03 ^ t04;
        t07 = p[a]    & w   ;
        t08 = t01 ^ w   ;
        t09 = p[b]    | t05;
        t10 =       ~ p[b]   ;
        x    = t08 ^ t09;
        t12 = t10 | t07;
        t13 = w     | x   ;
        z    = t02 ^ t12;
        t15 = t02 ^ t13;
        t16 = p[b]    ^ p[d]   ;
        y    = t16 ^ t15;
        k[kw]=w;
        k[kx]=x;
        k[ky]=y;
        k[kz]=z;
        }

/* S6:    7   2 12   5   8   4   6 11 14   9   1 15 13   3 10   0 */
/* depth = 8,3,6,3, Total gates=19 */
void RND06(int[] p, int a, int b, int c, int d, int[] k, int kw, int kx, int ky, int kz)
        { int t02, t03, t04, t05, t07, t08, t09, t10, t11, t12, t13, t15, t17, t18, t01;
        int w,x,y,z;
        t01 = p[a]    & p[d]   ;
        t02 = p[b]    ^ p[c]   ;
        t03 = p[a]    ^ p[d]   ;
        t04 = t01 ^ t02;
        t05 = p[b]    | p[c]   ;
        x    =       ~ t04;
        t07 = t03 & t05;
        t08 = p[b]    & x   ;
        t09 = p[a]    | p[c]   ;
```

```
        t10 = t07 ^ t08;
        t11 = p[b]    | p[d]   ;
        t12 = p[c]    ^ t11;
        t13 = t09 ^ t10;
        y    =        ~ t13;
        t15 = x    & t03;
        z    = t12 ^ t07;
        t17 = p[a]    ^ p[b]   ;
        t18 = y    ^ t15;
        w    = t17 ^ t18;
        k[kw]=w;
        k[kx]=x;
        k[ky]=y;
        k[kz]=z;
        }

/* InvS6:  15 10   1 13   5   3   6   0   4   9 14   7   2 12   8 11 */
/* depth = 5,3,8,6, Total gates=19 */
void InvRND06(int[] p, int a, int b, int c, int d, int[] k, int kw, int kx, int ky, int kz)
{ int t02, t03, t04, t05, t06, t07, t08, t09, t12, t13, t14, t15, t16, t17, t01;
        int w,x,y,z;
        t01 = p[a]    ^ p[c]   ;
        t02 =        ~ p[c]   ;
        t03 = p[b]    & t01;
        t04 = p[b]    | t02;
        t05 = p[d]    | t03;
        t06 = p[b]    ^ p[d]   ;
        t07 = p[a]    & t04;
        t08 = p[a]    | t02;
        t09 = t07 ^ t05;
        x    = t06 ^ t08;
        w    =        ~ t09;
        t12 = p[b]    & w   ;
        t13 = t01 & t05;
        t14 = t01 ^ t12;
        t15 = t07 ^ t13;
        t16 = p[d]    | t02;
        t17 = p[a]    ^ x   ;
        z    = t17 ^ t15;
        y    = t16 ^ t14;
        k[kw]=w;
        k[kx]=x;
        k[ky]=y;
        k[kz]=z;
}

/* S7:    1 13 15   0 14   8   2 11   7   4 12 10   9   3   5   6 */
/* depth = 10,7,10,4, Total gates=19 */
void RND07(int[] p, int a, int b, int c, int d, int[] k, int kw, int kx, int ky, int kz)
        { int t02, t03, t04, t05, t06, t08, t09, t10, t11, t13, t14, t15, t16, t17, t01;
        int w,x,y,z;
```

```
        t01 = p[a]     & p[c]   ;
        t02 =        ~ p[d]   ;
        t03 = p[a]     & t02;
        t04 = p[b]     | t01;
        t05 = p[a]     & p[b]   ;
        t06 = p[c]     ^ t04;
        z     = t03 ^ t06;
        t08 = p[c]     | z   ;
        t09 = p[d]     | t05;
        t10 = p[a]     ^ t08;
        t11 = t04 & z   ;
        x     = t09 ^ t10;
        t13 = p[b]     ^ x   ;
        t14 = t01 ^ x   ;
        t15 = p[c]     ^ t05;
        t16 = t11 | t13;
        t17 = t02 | t14;
        w     = t15 ^ t17;
        y     = p[a]     ^ t16;
        k[kw]=w;
        k[kx]=x;
        k[ky]=y;
        k[kz]=z;
        }

/* InvS7:    3   0   6 13   9 14 15   8   5 12 11   7 10   1   4   2 */
/* depth = 9,7,3,3, Total gates=18 */
void InvRND07(int[] p, int a, int b, int c, int d, int[] k, int kw, int kx, int ky, int kz)
        { int t02, t03, t04, t06, t07, t08, t09, t10, t11, t13, t14, t15, t16, t01;
        int w,x,y,z;
        t01 = p[a]     & p[b]   ;
        t02 = p[a]     | p[b]   ;
        t03 = p[c]     | t01;
        t04 = p[d]     & t02;
        z     = t03 ^ t04;
        t06 = p[b]     ^ t04;
        t07 = p[d]     ^ z   ;
        t08 =        ~ t07;
        t09 = t06 | t08;
        t10 = p[b]     ^ p[d]   ;
        t11 = p[a]     | p[d]   ;
        x     = p[a]     ^ t09;
        t13 = p[c]     ^ t06;
        t14 = p[c]     & t11;
        t15 = p[d]     | x   ;
        t16 = t01 | t10;
        w     = t13 ^ t15;
        y     = t14 ^ t16;
        k[kw]=w;
        k[kx]=x;
        k[ky]=y;
```

```
k[kz]=z;
}

    void RND08(int[] p, int a, int b, int c, int d, int[] k, int e, int f, int g, int h){ RND00(p,a,b,c,d,k,e,f,g,h);}
    void RND09(int[] p,int a, int b, int c, int d, int[] k, int e, int f, int g, int h){ RND01(p,a,b,c,d,k,e,f,g,h);}
    void RND10(int[] p,int a, int b, int c, int d, int[] k, int e, int f, int g, int h){ RND02(p,a,b,c,d,k,e,f,g,h);}
    void RND11(int[] p,int a, int b, int c, int d, int[] k, int e, int f, int g, int h){ RND03(p,a,b,c,d,k,e,f,g,h);}
    void RND12(int[] p,int a, int b, int c, int d, int[] k, int e, int f, int g, int h){ RND04(p,a,b,c,d,k,e,f,g,h);}
    void RND13(int[] p,int a, int b, int c, int d, int[] k, int e, int f, int g, int h){ RND05(p,a,b,c,d,k,e,f,g,h);}
    void RND14(int[] p,int a, int b, int c, int d, int[] k, int e, int f, int g, int h){ RND06(p,a,b,c,d,k,e,f,g,h);}
    void RND15(int[] p,int a, int b, int c, int d, int[] k, int e, int f, int g, int h){ RND07(p,a,b,c,d,k,e,f,g,h);}
    void RND16(int[] p,int a, int b, int c, int d, int[] k, int e, int f, int g, int h){ RND00(p,a,b,c,d,k,e,f,g,h);}
    void RND17(int[] p,int a, int b, int c, int d, int[] k, int e, int f, int g, int h){ RND01(p,a,b,c,d,k,e,f,g,h);}
    void RND18(int[] p,int a, int b, int c, int d, int[] k, int e, int f, int g, int h){ RND02(p,a,b,c,d,k,e,f,g,h);}
    void RND19(int[] p,int a, int b, int c, int d, int[] k, int e, int f, int g, int h){ RND03(p,a,b,c,d,k,e,f,g,h);}
    void RND20(int[] p,int a, int b, int c, int d, int[] k, int e, int f, int g, int h){ RND04(p,a,b,c,d,k,e,f,g,h);}
    void RND21(int[] p,int a, int b, int c, int d, int[] k, int e, int f, int g, int h){ RND05(p,a,b,c,d,k,e,f,g,h);}
    void RND22(int[] p,int a, int b, int c, int d, int[] k, int e, int f, int g, int h){ RND06(p,a,b,c,d,k,e,f,g,h);}
    void RND23(int[] p,int a, int b, int c, int d, int[] k, int e, int f, int g, int h){ RND07(p,a,b,c,d,k,e,f,g,h);}
    void RND24(int[] p,int a, int b, int c, int d, int[] k, int e, int f, int g, int h){ RND00(p,a,b,c,d,k,e,f,g,h);}
    void RND25(int[] p,int a, int b, int c, int d, int[] k, int e, int f, int g, int h){ RND01(p,a,b,c,d,k,e,f,g,h);}
    void RND26(int[] p,int a, int b, int c, int d, int[] k, int e, int f, int g, int h){ RND02(p,a,b,c,d,k,e,f,g,h);}
    void RND27(int[] p,int a, int b, int c, int d, int[] k, int e, int f, int g, int h){ RND03(p,a,b,c,d,k,e,f,g,h);}
    void RND28(int[] p,int a, int b, int c, int d, int[] k, int e, int f, int g, int h){ RND04(p,a,b,c,d,k,e,f,g,h);}
    void RND29(int[] p,int a, int b, int c, int d, int[] k, int e, int f, int g, int h){ RND05(p,a,b,c,d,k,e,f,g,h);}
    void RND30(int[] p,int a, int b, int c, int d, int[] k, int e, int f, int g, int h){ RND06(p,a,b,c,d,k,e,f,g,h);}
    void RND31(int[] p,int a, int b, int c, int d, int[] k, int e, int f, int g, int h){ RND07(p,a,b,c,d,k,e,f,g,h);}
```

```
        void InvRND08(int[] p,int a, int b, int c, int d,int[] k, int e, int f, int g, int
h){ InvRND00(p,a,b,c,d,k,e,f,g,h);}
        void InvRND09(int[] p,int a, int b, int c, int d,int[] k, int e, int f, int g, int
h){ InvRND01(p,a,b,c,d,k,e,f,g,h);}
        void InvRND10(int[] p,int a, int b, int c, int d,int[] k, int e, int f, int g, int
h){ InvRND02(p,a,b,c,d,k,e,f,g,h);}
        void InvRND11(int[] p,int a, int b, int c, int d,int[] k, int e, int f, int g, int
h){ InvRND03(p,a,b,c,d,k,e,f,g,h);}
        void InvRND12(int[] p,int a, int b, int c, int d,int[] k, int e, int f, int g, int
h){ InvRND04(p,a,b,c,d,k,e,f,g,h);}
        void InvRND13(int[] p,int a, int b, int c, int d,int[] k, int e, int f, int g, int
h){ InvRND05(p,a,b,c,d,k,e,f,g,h);}
        void InvRND14(int[] p,int a, int b, int c, int d,int[] k, int e, int f, int g, int
h){ InvRND06(p,a,b,c,d,k,e,f,g,h);}
        void InvRND15(int[] p,int a, int b, int c, int d,int[] k, int e, int f, int g, int
h){ InvRND07(p,a,b,c,d,k,e,f,g,h);}
        void InvRND16(int[] p,int a, int b, int c, int d,int[] k, int e, int f, int g, int
h){ InvRND00(p,a,b,c,d,k,e,f,g,h);}
        void InvRND17(int[] p,int a, int b, int c, int d,int[] k, int e, int f, int g, int
h){ InvRND01(p,a,b,c,d,k,e,f,g,h);}
        void InvRND18(int[] p,int a, int b, int c, int d,int[] k, int e, int f, int g, int
h){ InvRND02(p,a,b,c,d,k,e,f,g,h);}
        void InvRND19(int[] p,int a, int b, int c, int d,int[] k, int e, int f, int g, int
h){ InvRND03(p,a,b,c,d,k,e,f,g,h);}
        void InvRND20(int[] p,int a, int b, int c, int d,int[] k, int e, int f, int g, int
h){ InvRND04(p,a,b,c,d,k,e,f,g,h);}
        void InvRND21(int[] p,int a, int b, int c, int d,int[] k, int e, int f, int g, int
h){ InvRND05(p,a,b,c,d,k,e,f,g,h);}
        void InvRND22(int[] p,int a, int b, int c, int d,int[] k, int e, int f, int g, int
h){ InvRND06(p,a,b,c,d,k,e,f,g,h);}
        void InvRND23(int[] p,int a, int b, int c, int d,int[] k, int e, int f, int g, int
h){ InvRND07(p,a,b,c,d,k,e,f,g,h);}
        void InvRND24(int[] p,int a, int b, int c, int d,int[] k, int e, int f, int g, int
h){ InvRND00(p,a,b,c,d,k,e,f,g,h);}
        void InvRND25(int[] p,int a, int b, int c, int d,int[] k, int e, int f, int g, int
h){ InvRND01(p,a,b,c,d,k,e,f,g,h);}
        void InvRND26(int[] p,int a, int b, int c, int d,int[] k, int e, int f, int g, int
h){ InvRND02(p,a,b,c,d,k,e,f,g,h);}
        void InvRND27(int[] p,int a, int b, int c, int d,int[] k, int e, int f, int g, int
h){ InvRND03(p,a,b,c,d,k,e,f,g,h);}
        void InvRND28(int[] p,int a, int b, int c, int d,int[] k, int e, int f, int g, int
h){ InvRND04(p,a,b,c,d,k,e,f,g,h);}
        void InvRND29(int[] p,int a, int b, int c, int d,int[] k, int e, int f, int g, int
h){ InvRND05(p,a,b,c,d,k,e,f,g,h);}
        void InvRND30(int[] p,int a, int b, int c, int d,int[] k, int e, int f, int g, int
h){ InvRND06(p,a,b,c,d,k,e,f,g,h);}
        void InvRND31(int[] p,int a, int b, int c, int d,int[] k, int e, int f, int g, int
h){ InvRND07(p,a,b,c,d,k,e,f,g,h);}

        /* Linear transformations and key mixing: */
```

```
        void transform(int[] x, int x0, int x1, int x2, int x3, int[] y, int y0, int y1, int y2, int y3) {
                y[y0] = Integer.rotateLeft(x[x0], 13);
                y[y2] = Integer.rotateLeft(x[x2], 3);
                y[y1] = x[x1] ^ y[y0] ^ y[y2];
                y[y3] = x[x3] ^ y[y2] ^ ((int)y[y0])<<3;
                y[y1] = Integer.rotateLeft(y[y1], 1);
                y[y3] = Integer.rotateLeft(y[y3], 7);
                y[y0] = y[y0] ^ y[y1] ^ y[y3];
                y[y2] = y[y2] ^ y[y3] ^ ((int)y[y1]<<7);
                y[y0] = Integer.rotateLeft(y[y0], 5);
                y[y2] = Integer.rotateLeft(y[y2], 22);
        }

        void inv_transform(int[] x, int x0, int x1, int x2, int x3, int[] y, int y0, int y1, int y2, int y3) {
                y[y2] = Integer.rotateRight(x[x2], 22);
                y[y0] = Integer.rotateRight(x[x0], 5);
                y[y2] = y[y2] ^ x[x3] ^ ((int)x[x1]<<7);
                y[y0] = y[y0] ^ x[x1] ^ x[x3];
                y[y3] = Integer.rotateRight(x[x3], 7);
                y[y1] = Integer.rotateRight(x[x1], 1);
                y[y3] = y[y3] ^ y[y2] ^ ((int)y[y0])<<3;
                y[y1] = y[y1] ^ y[y0] ^ y[y2];
                y[y2] = Integer.rotateRight(y[y2], 3);
                y[y0] = Integer.rotateRight(y[y0], 13);
        }

        void keying(int[] x, int x0, int x1, int x2, int x3, int[] subkey){    x[x0]^=subkey[0];
x[x1]^=subkey[1];
                                        x[x2]^=subkey[2]; x[x3]^=subkey[3];
        }

                /* PHI: Constant used in the key schedule */
        /*
                #define PHI 0x9e3779b9L
        */

                ////////////////////////////////////
                /*   The functions   */
        /*   The functions   */
        int serpent_makeKey( keyInstance key, int direction, int keyLen,
                                        int[] keyMaterial)
        {
           int i,j;
           int[] w = new int[132];
           int[] k = new int[132];
           int rc;

           if((direction != 0) && (direction != 1)){
              return(-1);
           }
```

```
 if(keyLen>256 || keyLen<1)
    return -2;

 key.direction=direction;
 key.keyLen=keyLen;

 for(i =0; i<64+1; i++){
            key.keyMaterial[i] = keyMaterial[i];
 }

 rc=serpent_convert_from_string(keyLen, keyMaterial, key.key);
 if(rc<=0)
    return -2;

 for(i=0; i<keyLen/32; i++)
    w[i]=key.key[i];
 if(keyLen<256)
    w[i]=(key.key[i]&((1<<((keyLen&31)))-1))|(1<<((keyLen&31)));
 for(i++; i<8; i++)
    w[i]=0;
 for(i=8; i<16; i++)
    w[i]=Integer.rotateLeft((w[i-8]^w[i-5]^w[i-3]^w[i-1]^0x9e3779b9^(i-8)),11);
 for(i=0; i<8; i++)
    w[i]=w[i+8];
 for(i=8; i<132; i++)
    w[i]=Integer.rotateLeft((w[i-8]^w[i-5]^w[i-3]^w[i-1]^0x9e3779b9^i),11);

RND03(w,  0,  1,  2,   3, k,  0,  1,  2,  3);
RND02(w,  4,  5,  6,  7, k,  4,  5,  6,  7);
RND01(w, 8,  9, 10,  11, k,  8,  9, 10, 11);
RND00(w, 12,  13, 14, 15, k, 12, 13, 14, 15);
RND31(w, 16,  17,  18,  19, k, 16, 17, 18, 19);
RND30(w, 20, 21,  22,  23, k, 20, 21, 22, 23);
RND29(w, 24, 25, 26, 27, k, 24, 25, 26, 27);
RND28(w, 28, 29, 30, 31, k, 28, 29, 30, 31);
RND27(w, 32, 33, 34, 35, k, 32, 33, 34, 35);
RND26(w, 36, 37, 38, 39, k, 36, 37, 38, 39);
RND25(w, 40, 41, 42, 43, k, 40, 41, 42, 43);
RND24(w, 44, 45, 46, 47, k, 44, 45, 46, 47);
RND23(w, 48, 49, 50, 51, k, 48, 49, 50, 51);
RND22(w, 52, 53, 54, 55, k, 52, 53, 54, 55);
RND21(w, 56, 57, 58, 59, k, 56, 57, 58, 59);
RND20(w, 60, 61, 62, 63, k, 60, 61, 62, 63);
RND19(w, 64, 65, 66, 67, k, 64, 65, 66, 67);
RND18(w, 68, 69, 70, 71, k, 68, 69, 70, 71);
RND17(w, 72, 73, 74, 75, k, 72, 73, 74, 75);
RND16(w, 76, 77, 78, 79, k, 76, 77, 78, 79);
RND15(w, 80, 81, 82, 83, k, 80, 81, 82, 83);
RND14(w, 84, 85, 86, 87, k, 84, 85, 86, 87);
RND13(w, 88, 89, 90, 91, k, 88, 89, 90, 91);
```

```
  RND12(w, 92, 93, 94, 95, k, 92, 93, 94, 95);
  RND11(w, 96, 97, 98, 99, k, 96, 97, 98, 99);
  RND10(w,100, 101,102,103, k,100,101,102,103);
  RND09(w,104,105, 106,107, k,104,105,106,107);
  RND08(w,108,109,110,111, k,108,109,110,111);
  RND07(w,112,113,114,115, k,112,113,114,115);
  RND06(w,116,117,118,119, k,116,117,118,119);
  RND05(w,120,121,122,123, k,120,121,122,123);
  RND04(w,124,125,126,127, k,124,125,126,127);
  RND03(w,128, 129,130,131, k,128,129,130,131);

    for(i=0; i<=32; i++)
      for(j=0; j<4; j++)
        key.subkeys[i][j] = k[4*i+j];

    return(1);
}

int cipherInit( cipherInstance cipher, int mode, int[] IV)
{
//   int i;
    int rc;

    if((mode != 1) &&
       (mode != 2) &&
       (mode != 3))
      return -4;

    cipher.mode = mode;                 /* MODE_ECB, MODE_CBC, or MODE_CFB1 */
    cipher.blockSize=128;
    if(mode != 1)
      {
        rc=serpent_convert_from_string(cipher.blockSize, IV, cipher.IVi);
        for(int i=0; i<4; i++){
              cipher.IVb[i*4+0] = (byte)cipher.IVi[i];
              cipher.IVb[i*4+1] = (byte)(cipher.IVi[i]>>>8);
              cipher.IVb[i*4+2] = (byte)(cipher.IVi[i]>>>16);
              cipher.IVb[i*4+3] = (byte)(cipher.IVi[i]>>>24);
        }
        if(rc<=0)
          return -5;
      }

    return 1;
}



int blockEncrypt(cipherInstance cipher,
                 keyInstance key,
                 int[] input,
```

```
                    int inputLen,
                    int[] outBuffer)
{
  int[] t = new int[4];
  int[] u = new int[4];
  int   b,   n, i;
  int bit, bit0, ctBit, carry;

  /*
   * Note about optimization: the code becomes slower of the calls to
   * serpent_encrypt and serpent_decrypt are replaced by inlined code.
   * (tested on Pentium 133MMX)
   */

  switch(cipher.mode)
    {
    case 1:
        for(b=0 ; b<inputLen; b+=128 ){
            if(obmode != 3){
                u[0] = pbuf[oip+0];
                u[1] = pbuf[oip+1];
                u[2] = pbuf[oip+2];
                u[3] = pbuf[oip+3];
            }
            if(obmode == 3){
                u[0] = cipher.IVi[0];
            u[1] = cipher.IVi[1];
            u[2] = cipher.IVi[2];
            u[3] = cipher.IVi[3];
            }
            serpent_encrypt( u, 0, x, 0, key.subkeys);
            if(obmode != 3){
                cbuf[oip+0]=x[0];
                cbuf[oip+1]=x[1];
                cbuf[oip+2]=x[2];
                cbuf[oip+3]=x[3];
                for(i=0; i<4; i++){
                    cbufb[4*(oip+i)+0] = (byte)(cbuf[oip+i]);
                    cbufb[4*(oip+i)+1] = (byte)(cbuf[oip+i]>>>8);
                    cbufb[4*(oip+i)+2] = (byte)(cbuf[oip+i]>>>16);
                    cbufb[4*(oip+i)+3] = (byte)(cbuf[oip+i]>>>24);
                }
            }
            oip+=4;
        }
        return inputLen;

    case 2:
        t[0] = cipher.IVi[0];
        t[1] = cipher.IVi[1];
        t[2] = cipher.IVi[2];
```

```
        t[3] = cipher.IVi[3];
        for(b=0; b<inputLen; b+=128)
        {
                t[0] ^= pbuf[oip+0];
                t[1] ^= pbuf[oip+1];
                t[2] ^= pbuf[oip+2];
                t[3] ^= pbuf[oip+3];
                serpent_encrypt(t,0, t,0, key.subkeys);
                cbuf[oip+0] = t[0];
                cbuf[oip+1] = t[1];
                cbuf[oip+2] = t[2];
                cbuf[oip+3] = t[3];
                for(i=0; i<4; i++){
                        cbufb[4*(oip+i)+0] = (byte)(cbuf[oip+i]);
                        cbufb[4*(oip+i)+1] = (byte)(cbuf[oip+i]>>>8);
                        cbufb[4*(oip+i)+2] = (byte)(cbuf[oip+i]>>>16);
                        cbufb[4*(oip+i)+3] = (byte)(cbuf[oip+i]>>>24);
                }
                oip += 4;
        }
        cipher.IVi[0] = t[0];
        cipher.IVi[1] = t[1];
        cipher.IVi[2] = t[2];
        cipher.IVi[3] = t[3];

        return inputLen;

    case 3:
                cipher.mode = 1;  /* do encryption in ECB */
                obmode = 3;
                for (n=0;n<inputLen;n++)
                        {
                        blockEncrypt(cipher,key,cipher.IVi,128,x);
                        for(i=0; i<4; i++){
                                cipher.IVb[i*4+0] = (byte)cipher.IVi[i];
                                cipher.IVb[i*4+1] = (byte)(cipher.IVi[i]>>>8);
                                cipher.IVb[i*4+2] = (byte)(cipher.IVi[i]>>>16);
                                cipher.IVb[i*4+3] = (byte)(cipher.IVi[i]>>>24);
                        }

                        bit0    =    (0x80 >>> (n & 7));/* which bit position in byte */
                        ctBit =    ((pbufb[n/8] & bit0) ^ (((byte)(x[0]) & 0x80) >>> (n&7)));
                        cbufb[n/8] =    (byte) ((cbufb[n/8] & ~ bit0) | ctBit);
                        carry =    (ctBit >>> (7 - (n&7)));
                        for (i=128/8-1;i>=0;i--)
                                {
                                bit =    (cipher.IVb[i] >>> 7);          /* save next "carry" from
shift */
                                cipher.IVb[i] = (byte) ((cipher.IVb[i] << 1) ^ carry);
                                carry = bit;
                                }
```

```
                        int jj =0;
                        for(i=0; i*4<128/8; i++){
                                for (int p = 0; p < 4; p++) {
                                        int tmpi = (int)(cipher.IVb[i*4+3-p] & 0xff);
                                        if(tmpi < 0){
                                                cipher.IVb[i*4+3-p] ^= 0x80;
                                                tmpi = cipher.IVb[i*4+3-p];
                                                tmpi += 0x80;
                                        }
                                        jj = (jj << 8) | tmpi;
                                }
                                cipher.IVi[i] = jj;
                        }



                        }
                    cipher.mode = 3;  /* restore mode for next time */
                    obmode = 0;
                    return inputLen;

      default:
        return -5;
      }
}

int blockDecrypt(cipherInstance cipher,
                     keyInstance key,
                     int[] input,
                     int inputLen,
                     int[] outBuffer)
{
   int[] t = new int[4];
   int[] u = new int[4];
   int[] v = new int[4];
   int    b, n, i;
   int bit, bit0, ctBit, carry;

   switch(cipher.mode)
      {
      case 1:
        for(b=0; b<inputLen; b+=128){
            if(obmode != 3){
                    u[0]=cbuf[oip+0];
                    u[1]=cbuf[oip+1];
                    u[2]=cbuf[oip+2];
                    u[3]=cbuf[oip+3];
            }
            if(obmode==3){
                    u[0] = cipher.IVi[0];
                u[1] = cipher.IVi[1];
```

```
            u[2] = cipher.IVi[2];
            u[3] = cipher.IVi[3];
        }
        serpent_decrypt( u, 0, x, 0, key.subkeys);
        if(obmode != 3){
                pbuf[oip+0]=x[0];
                pbuf[oip+1]=x[1];
                pbuf[oip+2]=x[2];
                pbuf[oip+3]=x[3];
                for(i=0; i<4; i++){
                        pbufb[4*(oip+i)+0] = (byte)(pbuf[oip+i]);
                        pbufb[4*(oip+i)+1] = (byte)(pbuf[oip+i]>>>8);
                        pbufb[4*(oip+i)+2] = (byte)(pbuf[oip+i]>>>16);
                        pbufb[4*(oip+i)+3] = (byte)(pbuf[oip+i]>>>24);
                }
        }
        oip += 4;
    }
    return inputLen;

case 2:
    t[0] = cipher.IVi[0];
    t[1] = cipher.IVi[1];
    t[2] = cipher.IVi[2];
    t[3] = cipher.IVi[3];
    for(b=0; b<inputLen; b+=128)
    {
        u[0]=cbuf[oip+0];
        u[1]=cbuf[oip+1];
        u[2]=cbuf[oip+2];
        u[3]=cbuf[oip+3];
        serpent_decrypt(u,0, v, 0, key.subkeys);
        v[0] ^= t[0];
        v[1] ^= t[1];
        v[2] ^= t[2];
        v[3] ^= t[3];
        t[0] = u[0];
        t[1] = u[1];
        t[2] = u[2];
        t[3] = u[3];
        pbuf[oip+0]=v[0];
        pbuf[oip+1]=v[1];
        pbuf[oip+2]=v[2];
        pbuf[oip+3]=v[3];
        for(i=0; i<4; i++){
                pbufb[4*(oip+i)+0] = (byte)(pbuf[oip+i]);
                pbufb[4*(oip+i)+1] = (byte)(pbuf[oip+i]>>>8);
                pbufb[4*(oip+i)+2] = (byte)(pbuf[oip+i]>>>16);
                        pbufb[4*(oip+i)+3] = (byte)(pbuf[oip+i]>>>24);
        }
        oip += 4;
```

```
                }
                cipher.IVi[0] = t[0];
                cipher.IVi[1] = t[1];
                cipher.IVi[2] = t[2];
                cipher.IVi[3] = t[3];

                return inputLen;

            case 3://blockDecrypt
                        cipher.mode = 1; /* do encryption in ECB */
                        obmode = 3;
                        for (n=0;n<inputLen;n++)
                        {
                                blockEncrypt(cipher, key, cipher.IVi, 128, x);
                                for(i=0; i<4; i++){
                                        cipher.IVb[i*4+0] = (byte)cipher.IVi[i];
                                        cipher.IVb[i*4+1] = (byte)(cipher.IVi[i]>>>8);
                                        cipher.IVb[i*4+2] = (byte)(cipher.IVi[i]>>>16);
                                        cipher.IVb[i*4+3] = (byte)(cipher.IVi[i]>>>24);
                                }
                                bit0    =   (0x80 >>> (n & 7));
                                ctBit =   (cbufb[n/8] & bit0);
                                pbufb[n/8] =    (byte) ((pbufb[n/8] & ~ bit0) |
                                                                        (ctBit  ^  (((byte)(x[0])  &  0x80)
>>> (n&7))));
                                carry =    (ctBit >>> (7 - (n&7)));
                                for (i=128/8-1;i>=0;i--)
                                        {
                                        bit =    (cipher.IVb[i] >>> 7);          /* save next "carry" from
shift */
                                        cipher.IVb[i] =    (byte) ((cipher.IVb[i] << 1) ^ carry);
                                        carry = bit;
                                        }
                                int jj=0;
                                for(i=0; i*4<128/8; i++){
                                        for (int p = 0; p < 4; p++) {
                                                int tmpi = (int)(cipher.IVb[i*4+3-p] & 0xff);
                                                if(tmpi < 0){
                                                        cipher.IVb[i*4+3-p] ^= 0x80;
                                                        tmpi = cipher.IVb[i*4+3-p];
                                                        tmpi += 0x80;
                                                }
                                                jj = (jj << 8) | tmpi;
                                        }
                                        cipher.IVi[i] = jj;
                                }
                        }
                        cipher.mode = 3; /* restore mode for next time */
                        obmode = 0;
                        return inputLen;
```

```
      default:
         return -5;
      }
}

void serpent_encrypt(int[] plaintext, int ps,
                     int[] ciphertext, int cs,
                     int[][] subkeys)
{
   int x0=0, x1=1, x2=2, x3=3;
   int y0=0, y1=1, y2=2, y3=3;
   int[] y = new int[4];
   int[] x = new int[4];

   x[x0]=plaintext[ps+0];
   x[x1]=plaintext[ps+1];
   x[x2]=plaintext[ps+2];
   x[x3]=plaintext[ps+3];

   /* Start to encrypt the plaintext x */
   keying(x, x0, x1, x2, x3, subkeys[ 0]);
   RND00(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
   transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
   keying(x, x0, x1, x2, x3, subkeys[ 1]);
   RND01(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
   transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
   keying(x, x0, x1, x2, x3, subkeys[ 2]);
   RND02(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
   transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
   keying(x, x0, x1, x2, x3, subkeys[ 3]);
   RND03(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
   transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
   keying(x, x0, x1, x2, x3, subkeys[ 4]);
   RND04(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
   transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
   keying(x, x0, x1, x2, x3, subkeys[ 5]);
   RND05(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
   transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
   keying(x, x0, x1, x2, x3, subkeys[ 6]);
   RND06(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
   transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
   keying(x, x0, x1, x2, x3, subkeys[ 7]);
   RND07(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
   transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
   keying(x, x0, x1, x2, x3, subkeys[ 8]);
   RND08(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
   transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
   keying(x, x0, x1, x2, x3, subkeys[ 9]);
   RND09(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
   transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
   keying(x, x0, x1, x2, x3, subkeys[10]);
```

```
RND10(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
keying(x, x0, x1, x2, x3, subkeys[11]);
RND11(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
keying(x, x0, x1, x2, x3, subkeys[12]);
RND12(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
keying(x, x0, x1, x2, x3, subkeys[13]);
RND13(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
keying(x, x0, x1, x2, x3, subkeys[14]);
RND14(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
keying(x, x0, x1, x2, x3, subkeys[15]);
RND15(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
keying(x, x0, x1, x2, x3, subkeys[16]);
RND16(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
keying(x, x0, x1, x2, x3, subkeys[17]);
RND17(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
keying(x, x0, x1, x2, x3, subkeys[18]);
RND18(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
keying(x, x0, x1, x2, x3, subkeys[19]);
RND19(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
keying(x, x0, x1, x2, x3, subkeys[20]);
RND20(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
keying(x, x0, x1, x2, x3, subkeys[21]);
RND21(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
keying(x, x0, x1, x2, x3, subkeys[22]);
RND22(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
keying(x, x0, x1, x2, x3, subkeys[23]);
RND23(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
keying(x, x0, x1, x2, x3, subkeys[24]);
RND24(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
keying(x, x0, x1, x2, x3, subkeys[25]);
RND25(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
keying(x, x0, x1, x2, x3, subkeys[26]);
RND26(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
keying(x, x0, x1, x2, x3, subkeys[27]);
```

```
    RND27(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
    transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
    keying(x, x0, x1, x2, x3, subkeys[28]);
    RND28(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
    transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
    keying(x, x0, x1, x2, x3, subkeys[29]);
    RND29(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
    transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
    keying(x, x0, x1, x2, x3, subkeys[30]);
    RND30(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
    transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
    keying(x, x0, x1, x2, x3, subkeys[31]);
    RND31(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
    x[x0] = y[y0]; x[x1] = y[y1]; x[x2] = y[y2]; x[x3] = y[y3];
    keying(x, x0, x1, x2, x3, subkeys[32]);
    /* The ciphertext is now in x */

    ciphertext[cs+0] = x[x0];
    ciphertext[cs+1] = x[x1];
    ciphertext[cs+2] = x[x2];
    ciphertext[cs+3] = x[x3];
}

void serpent_decrypt(int[] ciphertext, int cs,
                     int[] plaintext, int ps,
                     int[][] subkeys)
{
    int x0=0, x1=1, x2=2, x3=3;
    int y0=0, y1=1, y2=2, y3=3;
    int[] x = new int[4];
    int[] y = new int[4];

    x[x0]=ciphertext[cs+0];
    x[x1]=ciphertext[cs+1];
    x[x2]=ciphertext[cs+2];
    x[x3]=ciphertext[cs+3];

    /* Start to decrypt the ciphertext x */
    keying(x, x0, x1, x2, x3, subkeys[32]);
    InvRND31(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
    keying(y, y0, y1, y2, y3, subkeys[31]);
    inv_transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
    InvRND30(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
    keying(y, y0, y1, y2, y3, subkeys[30]);
    inv_transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
    InvRND29(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
    keying(y, y0, y1, y2, y3, subkeys[29]);
    inv_transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
    InvRND28(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
    keying(y, y0, y1, y2, y3, subkeys[28]);
    inv_transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
```

```
InvRND27(x, x0, x1, x2, x3,y,    y0, y1, y2, y3);
keying(y, y0, y1, y2, y3, subkeys[27]);
inv_transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
InvRND26(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
keying(y, y0, y1, y2, y3, subkeys[26]);
inv_transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
InvRND25(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
keying(y, y0, y1, y2, y3, subkeys[25]);
inv_transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
InvRND24(x, x0, x1, x2, x3,y,    y0, y1, y2, y3);
keying(y, y0, y1, y2, y3, subkeys[24]);
inv_transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
InvRND23(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
keying(y, y0, y1, y2, y3, subkeys[23]);
inv_transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
InvRND22(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
keying(y, y0, y1, y2, y3, subkeys[22]);
inv_transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
InvRND21(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
keying(y, y0, y1, y2, y3, subkeys[21]);
inv_transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
InvRND20(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
keying(y, y0, y1, y2, y3, subkeys[20]);
inv_transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
InvRND19(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
keying(y, y0, y1, y2, y3, subkeys[19]);
inv_transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
InvRND18(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
keying(y, y0, y1, y2, y3, subkeys[18]);
inv_transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
InvRND17(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
keying(y, y0, y1, y2, y3, subkeys[17]);
inv_transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
InvRND16(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
keying(y, y0, y1, y2, y3, subkeys[16]);
inv_transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
InvRND15(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
keying(y, y0, y1, y2, y3, subkeys[15]);
inv_transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
InvRND14(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
keying(y, y0, y1, y2, y3, subkeys[14]);
inv_transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
InvRND13(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
keying(y, y0, y1, y2, y3, subkeys[13]);
inv_transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
InvRND12(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
keying(y, y0, y1, y2, y3, subkeys[12]);
inv_transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
InvRND11(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
keying(y, y0, y1, y2, y3, subkeys[11]);
inv_transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
```

```
    InvRND10(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
    keying(y, y0, y1, y2, y3, subkeys[10]);
    inv_transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
    InvRND09(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
    keying(y, y0, y1, y2, y3, subkeys[ 9]);
    inv_transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
    InvRND08(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
    keying(y, y0, y1, y2, y3, subkeys[ 8]);
    inv_transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
    InvRND07(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
    keying(y, y0, y1, y2, y3, subkeys[ 7]);
    inv_transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
    InvRND06(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
    keying(y, y0, y1, y2, y3, subkeys[ 6]);
    inv_transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
    InvRND05(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
    keying(y, y0, y1, y2, y3, subkeys[ 5]);
    inv_transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
    InvRND04(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
    keying(y, y0, y1, y2, y3, subkeys[ 4]);
    inv_transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
    InvRND03(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
    keying(y, y0, y1, y2, y3, subkeys[ 3]);
    inv_transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
    InvRND02(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
    keying(y, y0, y1, y2, y3, subkeys[ 2]);
    inv_transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
    InvRND01(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
    keying(y, y0, y1, y2, y3, subkeys[ 1]);
    inv_transform(y, y0, y1, y2, y3, x, x0, x1, x2, x3);
    InvRND00(x, x0, x1, x2, x3, y, y0, y1, y2, y3);
    x[x0] = y[y0]; x[x1] = y[y1]; x[x2] = y[y2]; x[x3] = y[y3];
    keying(x, x0, x1, x2, x3, subkeys[ 0]);
    /* The plaintext is now in x */

    plaintext[ps+0] = x[x0];
    plaintext[ps+1] = x[x1];
    plaintext[ps+2] = x[x2];
    plaintext[ps+3] = x[x3];
  }

//                #define min(x,y) (((x)<(y))?(x):(y))

  int serpent_convert_from_string(int len, int[] str, int[] val)
  /* the size of val must be at least the next multiple of 32 */
  /* bits after len bits */
  {
    int is, iv, i, j, k;
    byte[] tmpch4 = new byte[4];
    int tmpi =0, jj = 0;
    int slen = (((str.length)<((len+3)/4))?(str.length):((len+3)/4));// min(str.length, (len+3)/4);
```

```
        if(len<0)
          return -1;                /* Error!!! */

        if(len>slen*4 || len<slen*4-3)
          return -1;                /* Error!!! */

        for(is=0; is<slen; is++)
          if(((str[is]<'0')||(str[is]>'9')) &&
             ((str[is]<'A')||(str[is]>'F')) &&
             ((str[is]<'a')||(str[is]>'f')))
            return -1;  /* Error!!! */

        for(is=slen, iv=0; is>=8; is-=8, iv++)
          {
            byte t;
//          sscanf(&str[is-8], "%08lX", &t);
                for( i=0; i<4; i++){
                        j =    str[is-8+2*i];
                        k =    str[is-8+2*i+1];
                        if(j>=0x30 && j<=0x39) j =    (j-0x30);
                        else{
                                if(j>=0x41 && j<=0x46) j =    (j-0x41+0x0A);
                                if(j>=0x61 && j<=0x66) j =    (j-0x61+0x0A);
                        }
                        if(k>=0x30 && k<=0x39) k =    (k-0x30);
                        else{
                                if(k>=0x41 && k<=0x46) k =    (k-0x41+0x0A);
                                if(k>=0x61 && k<=0x66) k =    (k-0x61+0x0A);
                        }
                        tmpch4[i] =    (byte) (j*0x10 + k);
                }
                for (int p = 0; p < tmpch4.length; p++) {
                        tmpi = (int)(tmpch4[p] & 0xff);
                        if(tmpi < 0){
                                tmpch4[p] ^= 0x80;
                                tmpi = tmpch4[p];
                                tmpi += 0x80;
                        }
                        jj = (jj << 8) | tmpi;
                }
                        val[iv] = jj;
          }
        if(is>0)
        {
                byte[] tmp = new byte[10];
            byte t;
//          strncpy(tmp, str, is);
            for(i=0; i<is; i++){
                tmp[i] = (byte) str[i];
            }
```

```
                tmp[is] = 0;
//              sscanf(tmp, "%08lX", &t);
                    for( i=0; i<4; i++){
                            j =   str[is-8+2*i];
                            k =   str[is-8+2*i+1];
                            if(j>=0x30 && j<=0x39) j =   (j-0x30);
                            else{
                                    if(j>=0x41 && j<=0x46) j =   (j-0x41+0x0A);
                                    if(j>=0x61 && j<=0x66) j =   (j-0x61+0x0A);
                            }
                            if(k>=0x30 && k<=0x39) k =   (k-0x30);
                            else{
                                    if(k>=0x41 && k<=0x46) k =   (k-0x41+0x0A);
                                    if(k>=0x61 && k<=0x66) k =   (k-0x61+0x0A);
                            }
                            tmpch4[i] =   (byte) (j*0x10 + k);
                    }

                    tmpi =0; jj = 0;
                    for (int p = 0; p < tmpch4.length; p++) {
                              tmpi = (int)(tmpch4[p] & 0xff);
                              if(tmpi < 0){
                                        tmpch4[p] ^= 0x80;
                                        tmpi = tmpch4[p];
                                        tmpi += 0x80;
                              }
                              jj = (jj << 8) | tmpi;
                    }
                    val[iv++] = jj;
          }
          for(; iv<(len+31)/32; iv++)
             val[iv] = 0;
          return iv;
        }

        byte toChar( byte c )
        {
                if( c >= 0 && c <= 9 )                          // 0～9 ならば
                        return (byte) ( c + 0x30 );  // ASCII に変換して返す
                else if( c >= 10 && c <= 15 )           // 10～15 ならば
                        return (byte) ( c + 0x37 );  // A～F の ASCII を返す
                else
                        return ' ';
        }

        int[] serpent_convert_to_string(int len, int[] val, int[] str)
        /* str must have at least (len+3)/4+1 bytes. */
        {
                int i, j, k=0;
                byte[] tmp = new byte[10];
                if(len<0){
```

```
		str[0] = '0';
		return str;			/* Error!!! */
}

str[0] = 0;
i=len/32;
if((len&31)>0)
{
		//			byte[] tmp = new byte[10];
		//			sprintf(tmp, "%08lX", val[i]&(((len&31)<<1)-1));

		j = val[i]&(((len&31)<<1)-1);
		tmp[0] = (byte)(j);
		tmp[2] = (byte)(j>>>8);
		tmp[4] = (byte)(j>>>16);
		tmp[6] = (byte)(j>>>24);

		tmp[0] = (byte) (tmp[0]&0x0f);
		tmp[1] = (byte) (tmp[0]>>>4);
		tmp[2] = (byte)(tmp[2]&0x0f);
		tmp[3] = (byte)(tmp[2]>>>4);
		tmp[4] = (byte) (tmp[4]&0x0f);
		tmp[5] = (byte) (tmp[4]>>>4);
		tmp[6] = (byte)(tmp[6]&0x0f);
		tmp[7] = (byte)(tmp[6]>>>4);

		for(i=0; i<8 ; i++){
			tmp[i] = toChar(tmp[i]);
		}

		//			strcat(str, &tmp[8-(((len&31)+3)/4)]);

		k = 0;
		for(i=0; i<str.length; i++){
			if(str[i] == 0){k = i;}
		}
		for(i=0; i<(((len&31)+3)/4); i++){
			str[k+i] = tmp[i+8-(((len&31)+3)/4)];
		}
}
for(i--; i>=0; i--)
{

		//			sprintf(tmp, "%08lX", val[i]);
		//			strcat(str, tmp);

		j = val[i];
		tmp[0] = (byte)(j);
		tmp[2] = (byte)(j>>>8);
		tmp[4] = (byte)(j>>>16);
		tmp[6] = (byte)(j>>>24);
```

```
                tmp[0] = (byte) (tmp[0]&0x0f);
                tmp[1] = (byte) (tmp[0]>>>4);
                tmp[2] = (byte)(tmp[2]&0x0f);
                tmp[3] = (byte)(tmp[2]>>>4);
                tmp[4] = (byte) (tmp[4]&0x0f);
                tmp[5] = (byte) (tmp[4]>>>4);
                tmp[6] = (byte)(tmp[6]&0x0f);
                tmp[7] = (byte)(tmp[6]>>>4);

                for(i=0; i<8 ; i++){
                        tmp[i] = toChar(tmp[i]);
                }

                k = 0;
                for(i=0; i<8; i++){
                        if(str[i] == 0){
                                k = i;
                                break;
                        }
                }
        }
        for(i=0; i<(((len&31)+3)/4); i++){
                str[k+i] = tmp[i];
        }
        return str;
}


  ////////////////////////////////////////////////////////////////////////////////////
// SerpentEC.cpp : コンソール アプリケーション用のエントリ ポイントの定義
//
int s;
keyInstance keyI;
cipherInstance cipherI;


int SerpentEC(String keyfn, String ptfn, String ctfn)                    // 引数
へのポインタ
{
        File fkey, fin, fout;
    int len, rlen, blen4, blen;
        int mode,klen, rc=0;
        byte[] c_mode = new byte[3];
        byte[] c_klen = new byte[5];
        int[] c_key = new int[66];
        byte[] c_keyb = null;// new byte[66];
        int[] c_cini = new int[32+2];
        byte[] c_cinib = new byte[32+2];

        blen4 = 2048;
```

```
cipherInstance cipherI = new cipherInstance();
keyInstance keyI = new keyInstance();

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

fkey = new File(keyfn);
fkey.getParentFile().mkdir();
FileInputStream inkeyst=null;
try {
            inkeyst = new FileInputStream(fkey);
            inkeyst.read( c_mode);
            inkeyst.read( c_klen);
            klen = atoi(c_klen);
            c_keyb = new byte[klen/4+2];
            inkeyst.read( c_keyb );
            inkeyst.read( c_cinib);
} catch (IOException e5) {
            // TODO 自動生成された catch ブロック
            e5.printStackTrace();
}

fin = new File(ptfn);
fin.getParentFile().mkdir();
FileInputStream inptst=null;
try {
            inptst = new FileInputStream(fin);
} catch (FileNotFoundException e5) {
            // TODO 自動生成された catch ブロック
            e5.printStackTrace();
}

fout = new File(ctfn);
fout.getParentFile().mkdir();
FileOutputStream outctst=null;
try {
            outctst = new FileOutputStream(fout);
} catch (FileNotFoundException e5) {
            // TODO 自動生成された catch ブロック
            e5.printStackTrace();
}

mode = atoi(c_mode);
klen = atoi(c_klen);
blen = 128;

for(int i=0; i<32 ; i++){
            c_cini[i] = c_cinib[i];
}

if(klen<56 || 256<klen){
```

```
//                              printf("Wrong key size. ¥n");
                                return (-1);
                        }

                        /*Set mode*/
                        if(mode == 1){
                                int[] tmpb = new int[1];
                                tmpb[0] = ' ';
                                rc=cipherInit(cipherI, 1, tmpb);
                        }
                        if(mode == 2){
                                rc=cipherInit(cipherI, 2, c_cini);
                        }
                        if(mode == 3){
                                rc=cipherInit(cipherI, 3, c_cini);
                        }
                        if(rc<=0){
//                              printf("モード設定が出来ません。");
                                return(-2);
                        }

                        for(int i=0; i<klen/4;i++){
                                c_key[i] = c_keyb[i];
                        }
                        serpent_makeKey(keyI, 0,    klen, c_key );

                int flen;
                        try {
                                flen = inptst.available();
                             rlen = flen;

                // reset to start

                        s = 4;//sizeof(unsigned int);
                        // write the bytes of the file
//                      *((unsigned int*)pbuf) = rlen;
                        pbufb[0] = (byte) flen;
                        pbufb[1] = (byte)(flen>>>8);
                        pbufb[2] = (byte)(flen>>>16);
                        pbufb[3] = (byte)(flen>>>24);

                        if(flen > blen4-s){
                                len = inptst.read(pbufb, 4, blen4-s );
                        }else{
                                len = inptst.read(pbufb, 4, flen);
                        }
                rlen -= len;
                int jj =0;
                        for(int i=0; i*4<len+4; i++){
                                for(int k=0;k<4;k++){
                                        int tmp = pbufb[i*4+3-k];
```

```
                        if(tmp < 0){
                                byte tmpb = (byte) (pbufb[i*4+3-k] ^ 0x80);
                                tmp = tmpb;
                                tmp += 0x80;
                        }
                        jj = (jj << 8) | tmp;
                }
                pbuf[i] = jj;
                jj = 0;
        }

        int mesLength = len + 4;
        int block = mesLength/16 + 1;
        oip = 0;
        rc=blockEncrypt(cipherI, keyI, pbuf, 8*mesLength, cbuf);
outctst.write(cbufb, 0, blen4);

while(rlen > 0 && inptst.available()>0){
        // read a block and reduce the remaining byte count
        len = inptst.read(pbufb, 0, blen4);
        rlen -= len;
                rc=blockEncrypt(cipherI, keyI, pbuf, 8*blen4, cbuf);
        outctst.write(cbufb, 0, blen4);
}
        if(inkeyst != null)
        {
                try {
                        inkeyst.close();
                } catch (IOException e) {
                        // TODO 自動生成された catch ブロック
                        e.printStackTrace();
                }
        }

        if(outctst != null)
        {
                try {
                        outctst.close();
                } catch (IOException e) {
                        // TODO 自動生成された catch ブロック
                        e.printStackTrace();
                }
        }

        if(inptst != null)
        {
                try {
                        inptst.close();
                } catch (IOException e) {
                        // TODO 自動生成された catch ブロック
                        e.printStackTrace();
```

```
                }
            }
            } catch (IOException e1) {
                // TODO 自動生成された catch ブロック
                e1.printStackTrace();
            }
            return 0;
        }


        int atoi( byte s[] ) {
            int i, n, sign;

            for( i = 0; s[i] == ' ' ; i++ )  //先頭の空白を読み飛ばす
                ;
            sign = ( s[i] == '-' ) ? -1 : 1;      //符号を保存する
            if( s[i] == '-' || s[i] == '+' )      //符号を飛ばす
                i++;
            for( n = 0; i< s.length - 2 ; i++ )  //s[i]が数字のあいだ、n へ
                n = 10 * n + ( s[i] - '0' );
            return sign * n;                          //符号を反映
        }

    ////////////////////////////////////////////////////////
        void SerpentDC(String keyfn, String ctfn, String ptfn)                  // 引数へのポイ
ンタ
        {
            int i;
            byte[] c_mode = new byte[3];
            byte[] c_klen = new byte[5];
            int[] c_key = new int[64+2];
            byte[] c_keyb = null;//new byte[64+2];
            int[] c_cini = new int[32+2];
            byte[] c_cinib = new byte[32+2];

            int j,k;

          int len, rlen, blen4, pfilelen;
            int mode,klen,blen,rc=0;

            blen4 = 2048;

            cipherInstance cipherI = new cipherInstance();
            keyInstance keyI = new keyInstance();
            ////////////////////////////////////////////////////

            try{
            FileOutputStream foutst = openFileOutput(ptfn,MODE_PRIVATE);
            FileInputStream finst = openFileInput(ctfn);

            File fkey = new File(keyfn);
```

```
                fkey.getParentFile().mkdir();
                FileInputStream inkeyst=null;
                try {
                        inkeyst = new FileInputStream(fkey);
                        inkeyst.read(c_mode);
                        inkeyst.read(c_klen);
                        klen = atoi(c_klen);
                        c_keyb = new byte[klen/4+2];
                        inkeyst.read(c_keyb);
                        inkeyst.read(c_cinib);
                } catch (IOException e3) {
                        // TODO 自動生成された catch ブロック
                        e3.printStackTrace();
                }//127

                mode = atoi(c_mode);
                klen = atoi(c_klen);
                blen = 128;

                for(i=0; i<32 ; i++){
                        c_cini[i] = c_cinib[i];
                }

                if(klen<56 || 256<klen){
//                      printf("Wrong key size. ¥n");
                        return ;
                }

                /*Set mode*/
                if(mode == 1){
                        int[] tmpb = new int[1];
                        tmpb[0] = ' ';
                        rc=cipherInit(cipherI, 1, tmpb);
                }
                if(mode == 2){
                        rc=cipherInit(cipherI, 2, c_cini);
                }
                if(mode == 3){
                        rc=cipherInit(cipherI, 3, c_cini);
                }
                if(rc<=0){
//                      printf("モード設定が出来ません。");
                        return;
                }

                for(i=0; i<klen/4;i++){
                        c_key[i] = c_keyb[i];
                }
                serpent_makeKey(keyI, 1,    klen, c_key );

        int flen = finst.available();
```

```
rlen = flen;

        s = 4;//sizeof(unsigned long);
        // write the bytes of the file
        if(blen4<=flen){
                len = finst.read(cbufb, 0, blen4 );
        }else{
                len = finst.read(cbufb, 0, flen );
        }
        rlen = rlen - len;

        if(len < blen4){ return ;    }

        int jj =0;
        for(i=0; i*4<len; i++){
                for(int p=0; p<4; p++){
                        int tmp = cbufb[i*4+3-p];
                        if(tmp < 0){
                                byte tmpb = (byte) (cbufb[i*4+3-p] ^ 0x80);
                                tmp = tmpb;
                                tmp += 0x80;
                        }
                        jj = (jj << 8) | tmp;
                }
                cbuf[i] = jj;
                jj = 0;
        }

        ob=0; oip=0;
        rc=blockDecrypt(cipherI, keyI, cbuf, 8*flen, pbuf);
                // 復号文出力
//      pfilelen = *((long*)(pbuf));
        byte[] tmpch4 = new byte[4];
        tmpch4[0] = pbufb[0];//(byte) pbuf[0];
        tmpch4[1] = pbufb[1];//(byte) (pbuf[0]>>>8);
        tmpch4[2] = pbufb[2];//(byte) (pbuf[0]>>>16);
        tmpch4[3] = pbufb[3];//(byte) (pbuf[0]>>>24);
        jj = 0;
        int tmp = 0;
        for (int p = 0; p < tmpch4.length; p++) {
                tmp = (tmpch4[3-p] & 0xff);
                if(tmp < 0){
                        tmpch4[3-p] ^= 0x80;
                        tmp = tmpch4[3-p];
                        tmp += 0x80;
                }
                jj = (jj << 8) | tmp;
        }
        pfilelen = jj;

        if(pfilelen <= blen4 - s){
```

```
                foutst.write(pbufb, s, pfilelen);
                if(inkeyst != null)
                {
                        try {
                                inkeyst.close();
                        } catch (IOException e) {
                                // TODO 自動生成された catch ブロック
                                e.printStackTrace();
                        }
                }

                if(foutst != null)
                {
                        try {
                                foutst.close();
                        } catch (IOException e) {
                                // TODO 自動生成された catch ブロック
                                e.printStackTrace();
                        }
                }

                if(finst != null)
                {
                        try {
                                finst.close();
                        } catch (IOException e) {
                                // TODO 自動生成された catch ブロック
                                e.printStackTrace();
                        }
                }
                return;// 0;
        }
        else{
                foutst.write(pbufb, s, blen4 - s);
                pfilelen -= (blen4 - s);
        }

if((rlen <= blen4) && (rlen > 0))
        {       // if the file length is less than or equal to 2048 bytes
        len = finst.read(cbufb, 1, blen4 );
        rlen -= len;
        if(rlen > 0){ return; }
                rc=blockDecrypt(cipherI, keyI, cbuf, 8*blen4, pbuf);
                foutst.write(pbufb, 1, pfilelen) ;
                if(inkeyst != null)
                {
                        try {
                                inkeyst.close();
                        } catch (IOException e) {
                                // TODO 自動生成された catch ブロック
                                e.printStackTrace();
```

```
                        }
                    }

                    if(foutst != null)
                    {
                        try {
                            foutst.close();
                        } catch (IOException e) {
                            // TODO 自動生成された catch ブロック
                            e.printStackTrace();
                        }
                    }

                    if(finst != null)
                    {
                        try {
                            finst.close();
                        } catch (IOException e) {
                            // TODO 自動生成された catch ブロック
                            e.printStackTrace();
                        }
                    }
                    return;// 0;
                }
                else
                {    // if the file length is more 1024 bytes
                     // read the file a block at a time
                     while(rlen > 0 && finst.available()>0)
                     {
                          // read a block and reduce the remaining byte count
                          len = finst.read(cbufb, 1, blen4);
                          rlen -= len;
                          if((rlen>0) && (len==blen4)){
                                                    rc=blockDecrypt(cipherI,  keyI,  cbuf,  8*blen4,
pbuf);
                                                    foutst.write(pbufb, 1, blen4);
                                                    pfilelen -= blen4;
                                    }
                          if(rlen<=0){
                                                    rc=blockDecrypt(cipherI,  keyI,  cbuf,  8*blen4,
pbuf);
                                                    foutst.write(pbufb, 1, pfilelen);
                                                    if(inkeyst != null)
                                                    {
                                                        try {
                                                            inkeyst.close();
                                                        } catch (IOException e) {
                                                            // TODO 自動生成された catch
ブロック
                                                            e.printStackTrace();
                                                        }
```

```
                                        }

                                        if(foutst != null)
                                        {
                                                try {
                                                        foutst.close();
                                                } catch (IOException e) {
                                                        // TODO 自動生成された catch
ブロック
                                                        e.printStackTrace();
                                                }
                                        }

                                        if(finst != null)
                                        {
                                                try {
                                                        finst.close();
                                                } catch (IOException e) {
                                                        // TODO 自動生成された catch
ブロック
                                                        e.printStackTrace();
                                                }
                                        }
                                        return;
                                }
                        }
                }
                        if(inkeyst != null)
                        {
                                try {
                                        inkeyst.close();
                                } catch (IOException e) {
                                        // TODO 自動生成された catch ブロック
                                        e.printStackTrace();
                                }
                        }

                        if(foutst != null)
                        {
                                try {
                                        foutst.close();
                                } catch (IOException e) {
                                        // TODO 自動生成された catch ブロック
                                        e.printStackTrace();
                                }
                        }

                        if(finst != null)
                        {
                                try {
                                        finst.close();
```

```
                                } catch (IOException e) {
                                        // TODO 自動生成された catch ブロック
                                        e.printStackTrace();
                                }
                        }

                        return;// 0;

                }catch (IOException e) {
                        // TODO 自動生成された catch ブロック
                        e.printStackTrace();
                }

                }

}
package yu.com.pcs.jp.sumaho.cg5mail;

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

import android.R.bool;
import android.app.Activity;
import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.os.Bundle;

import java.io.BufferedInputStream;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;

import yu.com.pcs.jp.sumaho.cg5mail.MARSActivity.MARScipherInstance;
import yu.com.pcs.jp.sumaho.cg5mail.MARSActivity.MARSkeyInstance;
import yu.com.pcs.jp.sumaho.cg5mail.MailViewActivity.cipherInstance;
import yu.com.pcs.jp.sumaho.cg5mail.MailViewActivity.keyInstance;

public class TwofishActivity extends Activity{
        String prgn;
        String keyfn;
        String inputfn;
        String outputfn;
```

```
        @Override
        protected void onCreate(Bundle savedInstanceState) {
           super.onCreate(savedInstanceState);
                  prgn = "";
              keyfn = "";
              inputfn = "";
              outputfn = "";
        }

////////////////////////////////////////////////////////////////////////////
////////////////////// Twofish EC DC  ////////////////////////////////////////////


        /*****************************************************************************
        PLATFORM.H    -- Platform-specific defines for TWOFISH code

        Submitters:
                Bruce Schneier, Counterpane Systems
                Doug Whiting,   Hi/fn
                John Kelsey,    Counterpane Systems
                Chris Hall,             Counterpane Systems
                David Wagner,   UC Berkeley

        Code Author:            Doug Whiting,   Hi/fn

        Version  1.00           April 1998

        Copyright 1998, Hi/fn and Counterpane Systems.  All rights reserved.

        Notes:
                *       Tab size is set to 4 characters in this file

*****************************************************************************/

/* use intrinsic rotate if possible */
int     ROL(int x, int n){return (((x) << ((n) & 0x1F)) | ((x) >> (32-((n) & 0x1F))));}
int     ROR(int x, int n){return (((x) >> ((n) & 0x1F)) | ((x) << (32-((n) & 0x1F))));}

/*
if(0!= _MSC_VER){
#include<stdlib.h>                                      // get prototypes for rotation functions
#undef  ROL
#undef  ROR
#pragma intrinsic(_lrotl,_lrotr)                // use intrinsic compiler rotations
#define ROL(x,n)        _lrotl(x,n)
#define ROR(x,n)        _lrotr(x,n)
#endif
}
*/
```

```
/*
#ifdef _M_IX86
#define		LittleEndian		1		// e.g., 1 for Pentium, 0 for 68K
#define		ALIGN32				0		// need dword alignment? (no for Pentium)
#else	// non-Intel platforms
*/


//#if LittleEndian
int Bswap(int x){return		(x);}		/* NOP for little-endian machines */
int		ADDR_XOR	=		0;		/* NOP for little-endian machines */
//#endif

/*	Macros for extracting bytes from dwords (correct for endianness) */
/*
int	_b(int[] x, int N){return	(x[((N) & 3)] ^ ADDR_XOR) ;}// pick bytes out of a dword

int	b0(int[] x){return		_b(x,0);} // extract LSB of DWORD
int	b1(int[] x){return		_b(x,1);}
int	b2(int[] x){return		_b(x,2);}
int	b3(int[] x){return		_b(x,3);}	// extract MSB of DWORD
*/


	////////////////////////////////////////////////////////////////////////////
	/* aes.h */

	/* ---------- See examples at end of this file for typical usage -------- */

	/* AES Cipher header file for ANSI C Submissions
		Lawrence E. Bassham III
		Computer Security Division
		National Institute of Standards and Technology

		This sample is to assist implementers developing to the
	Cryptographic API Profile for AES Candidate Algorithm Submissions.
	Please consult this document as a cross-reference.

		ANY CHANGES, WHERE APPROPRIATE, TO INFORMATION PROVIDED IN THIS FILE
	MUST BE DOCUMENTED. CHANGES ARE ONLY APPROPRIATE WHERE SPECIFIED WITH
	THE STRING "CHANGE POSSIBLE". FUNCTION CALLS AND THEIR PARAMETERS
	CANNOT BE CHANGED. STRUCTURES CAN BE ALTERED TO ALLOW IMPLEMENTERS TO
	INCLUDE IMPLEMENTATION SPECIFIC INFORMATION.
	*/
```

```
        /* Includes:
                Standard include files
        */

//      #include<stdio.h>
//      #include"platform.h"                    /* platform-specific defines */

        /*      Defines:
                        Add any additional defines you need
        */

        int     DIR_ENCRYPT=0;                  /* Are we encrpyting? */
        int     DIR_DECRYPT =       1 ;             /* Are we decrpyting? */
        int     MODE_ECB    =       1 ;             /* Are we ciphering in ECB mode? */
        int     MODE_CBC    =       2 ;             /* Are we ciphering in CBC mode? */
        int     MODE_CFB1 =         3 ;             /* Are we ciphering in 1-bit CFB mode? */

        int     TRUE            =       1;
        int     FALSE           =       0;

        int     BAD_KEY_DIR  =      -1;     /* Key direction is invalid (unknown value) */
        int     BAD_KEY_MAT         =       -2;     /* Key material not of correct length */
        int     BAD_KEY_INSTANCE = -3;      /* Key passed is not valid */
        int     BAD_CIPHER_MODE =  -4;      /* Params struct passed to cipherInit invalid */
        int     BAD_CIPHER_STATE =  -5;     /* Cipher in wrong state (e.g., not initialized) */

        /* CHANGE POSSIBLE: inclusion of algorithm specific defines */
        /* TWOFISH specific definitions */
        int             MAX_KEY_SIZE=       64;     /* # of ASCII chars needed to represent a
key */
        int             MAX_IV_SIZE         =       16;     /* # of bytes needed to represent
an IV */
        int             BAD_INPUT_LEN       =       -6;     /* inputLen not a multiple of
block size */
        int             BAD_PARAMS          =       -7;     /* invalid parameters */
        int             BAD_IV_MAT          =       -8;     /* invalid IV text */
        int             BAD_ENDIAN          =       -9;     /* incorrect endianness define */
        int             BAD_ALIGN32         =       -10     ;/* incorrect 32-bit alignment */

        int             BLOCK_SIZE          =       128;    /* number of bits per block */
        int             MAX_ROUNDS          =        16;    /* max # rounds (for allocating
subkey array) */
        int             ROUNDS_128          =        16;    /* default number of rounds for
128-bit keys*/
        int             ROUNDS_192          =        16;    /* default number of rounds for
192-bit keys*/
        int             ROUNDS_256  =                16;    /* default number of rounds for
256-bit keys*/
        int             MAX_KEY_BITS        =       256;    /* max number of bits of key */
        int             MIN_KEY_BITS =      128;    /* min number of bits of key (zero pad) */
        int     VALID_SIG       = 0x48534946;   /* initialization signature ('FISH') */
```

```
        int             MCT_OUTER   =           400     ;/* MCT outer loop */
        int             MCT_INNER   =       10000;      /* MCT inner loop */
        int     REENTRANT           =       1       ;/* nonzero forces reentrant code (slightly
slower) */

        int             INPUT_WHITEN        =   0       ;/* subkey array indices */
        int     OUTPUT_WHITEN       =   ( INPUT_WHITEN + BLOCK_SIZE/32);
        int             ROUND_SUBKEYS       =   (OUTPUT_WHITEN                  +
BLOCK_SIZE/32);     /* use 2 * (# rounds) */
        int             TOTAL_SUBKEYS       =   (ROUND_SUBKEYS                  +
2*MAX_ROUNDS);

        /* Typedefs:
            Typedef'ed data storage elements. Add any algorithm specific
            parameters at the bottom of the structs as appropriate.
        */

//      typedef unsigned char BYTE;
//      typedef  unsigned long DWORD;           /* 32-bit unsigned quantity */
//      typedef DWORD fullSbox[4][256];

        /* The structure for key information */
//      typedef struct
        class keyInstance       {
            int direction;                                  /* Key used for encrypting or
decrypting? */

            int  keyLen;                                    /* Length of the key */
            byte[] keyMaterial = new byte[MAX_KEY_SIZE+4];/* Raw key data in ASCII */

            /* Twofish-specific parameters: */
            int keySig;                                     /* set to VALID_SIG by
makeKey() */
            int         numRounds;                          /* number of rounds in cipher */
            int[] key32 = new int[MAX_KEY_BITS/32];/* actual key bits, in dwords */
            int[] sboxKeys = new int[MAX_KEY_BITS/64];/* key bits used for S-boxes */
            int[] subKeys = new int[TOTAL_SUBKEYS];     /* round subkeys, input/output
whitening bits */
//      #if REENTRANT
            int[][] sBox8x32;//fullSbox sBox8x32;                   /*          fully
expanded S-box */
        //      #endif
            }

        /* The structure for cipher information */
//      typedef struct
        class cipherInstance{
            int  mode;                                      /*          MODE_ECB,
MODE_CBC, or MODE_CFB1 */
//      #if ALIGN32
            byte[] dummyAlign = new byte[3];                /* keep 32-bit alignment
```

```
*/
        //#endif
                byte[]  IV = new byte[MAX_IV_SIZE];                     /* CFB1 iv bytes  (CBC
uses iv32) */

                /* Twofish-specific parameters: */
                int cipherSig;                                  /* set to VALID_SIG by cipherInit() */
                int[] iv32 = new int[BLOCK_SIZE/32];            /* CBC IV bytes arranged as
dwords */
                }

        /* Function protoypes */
//      int makeKey(keyInstance *key, BYTE direction, int keyLen, char *keyMaterial);

//      int cipherInit(cipherInstance *cipher, BYTE mode, char *IV);

//      int blockEncrypt(cipherInstance *cipher, keyInstance *key, BYTE *input,
//                                      int inputLen, BYTE *outBuffer);

//      int blockDecrypt(cipherInstance *cipher, keyInstance *key, BYTE *input,
//                                      int inputLen, BYTE *outBuffer);

//      int     reKey(keyInstance *key); /* do key schedule using modified key.keyDwords */

        /* API to check table usage, for use in ECB_TBL KAT */
        int             TAB_DISABLE  =          0;
        int             TAB_ENABLE   =          1;
        int             TAB_RESET    =          2;
        int             TAB_QUERY    =          3;
        int             TAB_MIN_QUERY       =       50;
//      int TableOp(int op);


//      #define         CONST                   /* helpful C++ syntax sugar, NOP for
ANSI C */

        //#if BLOCK_SIZE == 128                 /* optimize block copies */
        void            Copy1(int[] d, int[] s, int N){     d[N] = s[N];}
        void            BlockCopy(int[]                 d,              int[]           s)
        { Copy1(d,s,0);Copy1(d,s,1);Copy1(d,s,2);Copy1(d,s,3); }
//      #else
///     #define         BlockCopy(d,s)  { memcpy(d,s,BLOCK_SIZE/8); }
///     #endif




        ////////////////////////////////////////////////////////////////////////////
        /**************************************************************************
        TABLE.H         -- Tables, macros, constants for Twofish S-boxes and MDS matrix
```

```
        Submitters:
                Bruce Schneier, Counterpane Systems
                Doug Whiting,   Hi/fn
                John Kelsey,    Counterpane Systems
                Chris Hall,             Counterpane Systems
                David Wagner,   UC Berkeley

        Code Author:            Doug Whiting,   Hi/fn

        Version  1.00           April 1998

        Copyright 1998, Hi/fn and Counterpane Systems.  All rights reserved.

        Notes:
                *       Tab size is set to 4 characters in this file
                *       These definitions should be used in optimized and unoptimized
                        versions to insure consistency.

***************************************************************************/

/* for computing subkeys */
int     SK_STEP                 =       0x02020202;
int     SK_BUMP         =               0x01010101;
int     SK_ROTL         =               9;

/* Reed-Solomon code parameters: (12,8) reversible code
        g(x) = x**4 + (a + 1/a) x**3 + a x**2 + (a + 1/a) x + 1
   where a = primitive root of field generator 0x14D */
int     RS_GF_FDBK      =       0x14D;          /* field generator */
void    RS_rem(int x)
        {
        int tmp1=0;
        int tmp2=0;
        byte    b    = (byte) (x >>> 24);
        if((b & 0x80)!=0){tmp1 = RS_GF_FDBK;}
        int g2 = ((b << 1) ^ tmp1 ) & 0xFF;
        if((b & 1)!=0){tmp2 = RS_GF_FDBK >>> 1;}
        int g3 = ((b >>> 1) & 0x7F) ^ tmp2    ^ g2 ;
           x = (x << 8) ^ (g3 << 24) ^ (g2 << 16) ^ (g3 << 8) ^ b;
        }

/*      Macros for the MDS matrix
*       The MDS matrix is (using primitive polynomial 169):
*       01   EF   5B   5B
*       5B   EF   EF   01
*       EF   5B   01   EF
*       EF   01   EF   5B
*-----------------------------------------------------------------
* More statistical properties of this matrix (from MDS.EXE output):
*
```

```
* Min Hamming weight (one byte difference) =  8. Max=26.  Total =  1020.
* Prob[8]:       7     23     42     20     52     95     88     94    121    128     91
*               102     76     41     24      8      4      1      3      0      0      0
* Runs[8]:       2      4      5      6      7      8      9     11
* MSBs[8]:       1      4     15      8     18     38     40     43
* HW= 8: 05040705 0A080E0A 14101C14 28203828 50407050 01499101 A080E0A0
* HW= 9: 04050707 080A0E0E 10141C1C 20283838 40507070 80A0E0E0 C6432020 07070504
*        0E0E0A08 1C1C1410 38382820 70705040 E0E0A080 202043C6 05070407 0A0E080E
*        141C101C 28382038 50704070 A0E080E0 4320C620 02924B02 089A4508
* Min Hamming weight (two byte difference) =  3. Max=28.  Total = 390150.
* Prob[3]:       7     18     55    149    270    914   2185   5761 11363 20719 32079
*             43492 51612 53851 52098 42015 31117 20854 11538   6223   2492   1033
* MDS OK, ROR:   6+   7+   8+   9+ 10+ 11+ 12+ 13+ 14+ 15+ 16+
*                17+ 18+ 19+ 20+ 21+ 22+ 23+ 24+ 25+ 26+
*/
int     MDS_GF_FDBK =           0x169;   /* primitive polynomial for GF(256)*/
int     LFSR1(int x) {
                if( ((x >> 1)   ^ (x & 0x01))!=0){return(MDS_GF_FDBK/2);}
                else{return 0;}
}
//              ((x) >> 1)   ^ (((x) & 0x01) ?     MDS_GF_FDBK/2 : 0))
int     LFSR2(int x){
        int tmp1 =0;
        int tmp2 = 0;
        if(((x) & 0x01) != 0){tmp2 = MDS_GF_FDBK/4;}
        if( ((x >>> 2)   ^ (x & 0x02)) != 0){  tmp1 = MDS_GF_FDBK/2;          }
        return(tmp1 ^ tmp2);
}
//      ( ((x) >> 2)   ^ (((x) & 0x02) ?     MDS_GF_FDBK/2 : 0)   ¥
//                                                                  ^       (((x)       &       0x01)       ?
MDS_GF_FDBK/4 : 0))

int     Mx_1(int x){return ((int)   (x));}              /* force result to dword so << will work */
int     Mx_X(int x){return ((int) ((x) ^                LFSR2(x)));}    /* 5B */
int     Mx_Y(int x){return ((int) ((x) ^ LFSR1(x) ^ LFSR2(x)));}    /* EF */

int     M00(int x){return       Mul_1(x);}
int     M01(int x){return       Mul_Y(x);}
int     M02(int x){return       Mul_X(x);}
int     M03(int x){return       Mul_X(x);}

int     M10(int x){return       Mul_X(x);}
int     M11     (int x){return  Mul_Y(x);}
int     M12     (int x){return  Mul_Y(x);}
int     M13(int x){return       Mul_1(x);}

int     M20(int x){return       Mul_Y(x);}
int     M21     (int x){return  Mul_X(x);}
int     M22     (int x){return  Mul_1(x);}
int     M23     (int x){return  Mul_Y(x);}
```

```
int     M30     (int x){return     Mul_Y(x);}
int     M31     (int x){return     Mul_1(x);}
int     M32     (int x){return     Mul_Y(x);}
int     M33     (int x){return     Mul_X(x);}

int     Mul_1(int x){return        Mx_1(x);}
int     Mul_X(int x){return        Mx_X(x);}
int     Mul_Y(int x){return        Mx_Y(x);}

/*      Define the fixed p0/p1 permutations used in keyed S-box lookup.
        By changing the following constant definitions for P_ij, the S-boxes will
        automatically get changed in all the Twofish source code. Note that P_i0 is
        the "outermost" 8x8 permutation applied.    See the f32() function to see
        how these constants are to be    used.
*/
int     P_00=   1;                                          /* "outermost" permutation */
int     P_01=   0;
int     P_02=   0;
int     P_03=   (P_01^1);                           /* "extend" to larger key sizes */
int     P_04=   1;

int     P_10=   0;
int     P_11=   0;
int     P_12=   1;
int     P_13=   (P_11^1);
int     P_14=   0;

int     P_20=   1;
int     P_21=   1;
int     P_22=   0;
int     P_23=   (P_21^1);
int     P_24=   0;

int     P_30=   0;
int     P_31=   1;
int     P_32=   1;
int     P_33=   (P_31^1);
int     P_34=   1;

int     p8(int P_ , int N){return   P8x8[P_ ][ N];}                 /* some syntax shorthand */

/* fixed 8x8 permutation S-boxes */

/*******************************************************************
*   07:07:14   05/30/98   [4x4]   TestCnt=256. keySize=128. CRC=4BD14D9E.
* maxKeyed:   dpMax = 18. lpMax =100. fixPt =   8. skXor =   0. skDup =   6.
* log2(dpMax[ 6..18])=     --- 15.42   1.33   0.89   4.05   7.98 12.05
* log2(lpMax[ 7..12])=   9.32   1.01   1.16   4.23   8.02 12.45
* log2(fixPt[ 0.. 8])=   1.44   1.44   2.44   4.06   6.01   8.21 11.07 14.09 17.00
* log2(skXor[ 0.. 0])
* log2(skDup[ 0.. 6])=     ---   2.37   0.44   3.94   8.36 13.04 17.99
```

```
*****************************************************************/
int P8x8[][]=
        {
/*   p0:    */
/*   dpMax       = 10.   lpMax       = 64.   cycleCnt=    1   1   1   0.            */
/* 817D6F320B59ECA4.ECB81235F4A6709D.BA5E6D90C8F32471.D7F4126E9B3085CA. */
/* Karnaugh maps:
*   0111 0001 0011 1010. 0001 1001 1100 1111. 1001 1110 0011 1110. 1101 0101 1111 1001.
*   0101 1111 1100 0100. 1011 0101 0010 0000. 0101 1000 1100 0101. 1000 0111 0011 0010.
*   0000 1001 1110 1101. 1011 1000 1010 0011. 0011 1001 0101 0000. 0100 0010 0101 1011.
*   0111 0100 0001 0110. 1000 1011 1110 1001. 0011 0011 1001 1101. 1101 0101 0000 1100.
*/
        {
        0xA9, 0x67, 0xB3, 0xE8, 0x04, 0xFD, 0xA3, 0x76,
        0x9A, 0x92, 0x80, 0x78, 0xE4, 0xDD, 0xD1, 0x38,
        0x0D, 0xC6, 0x35, 0x98, 0x18, 0xF7, 0xEC, 0x6C,
        0x43, 0x75, 0x37, 0x26, 0xFA, 0x13, 0x94, 0x48,
        0xF2, 0xD0, 0x8B, 0x30, 0x84, 0x54, 0xDF, 0x23,
        0x19, 0x5B, 0x3D, 0x59, 0xF3, 0xAE, 0xA2, 0x82,
        0x63, 0x01, 0x83, 0x2E, 0xD9, 0x51, 0x9B, 0x7C,
        0xA6, 0xEB, 0xA5, 0xBE, 0x16, 0x0C, 0xE3, 0x61,
        0xC0, 0x8C, 0x3A, 0xF5, 0x73, 0x2C, 0x25, 0x0B,
        0xBB, 0x4E, 0x89, 0x6B, 0x53, 0x6A, 0xB4, 0xF1,
        0xE1, 0xE6, 0xBD, 0x45, 0xE2, 0xF4, 0xB6, 0x66,
        0xCC, 0x95, 0x03, 0x56, 0xD4, 0x1C, 0x1E, 0xD7,
        0xFB, 0xC3, 0x8E, 0xB5, 0xE9, 0xCF, 0xBF, 0xBA,
        0xEA, 0x77, 0x39, 0xAF, 0x33, 0xC9, 0x62, 0x71,
        0x81, 0x79, 0x09, 0xAD, 0x24, 0xCD, 0xF9, 0xD8,
        0xE5, 0xC5, 0xB9, 0x4D, 0x44, 0x08, 0x86, 0xE7,
        0xA1, 0x1D, 0xAA, 0xED, 0x06, 0x70, 0xB2, 0xD2,
        0x41, 0x7B, 0xA0, 0x11, 0x31, 0xC2, 0x27, 0x90,
        0x20, 0xF6, 0x60, 0xFF, 0x96, 0x5C, 0xB1, 0xAB,
        0x9E, 0x9C, 0x52, 0x1B, 0x5F, 0x93, 0x0A, 0xEF,
        0x91, 0x85, 0x49, 0xEE, 0x2D, 0x4F, 0x8F, 0x3B,
        0x47, 0x87, 0x6D, 0x46, 0xD6, 0x3E, 0x69, 0x64,
        0x2A, 0xCE, 0xCB, 0x2F, 0xFC, 0x97, 0x05, 0x7A,
        0xAC, 0x7F, 0xD5, 0x1A, 0x4B, 0x0E, 0xA7, 0x5A,
        0x28, 0x14, 0x3F, 0x29, 0x88, 0x3C, 0x4C, 0x02,
        0xB8, 0xDA, 0xB0, 0x17, 0x55, 0x1F, 0x8A, 0x7D,
        0x57, 0xC7, 0x8D, 0x74, 0xB7, 0xC4, 0x9F, 0x72,
        0x7E, 0x15, 0x22, 0x12, 0x58, 0x07, 0x99, 0x34,
        0x6E, 0x50, 0xDE, 0x68, 0x65, 0xBC, 0xDB, 0xF8,
        0xC8, 0xA8, 0x2B, 0x40, 0xDC, 0xFE, 0x32, 0xA4,
        0xCA, 0x10, 0x21, 0xF0, 0xD3, 0x5D, 0x0F, 0x00,
        0x6F, 0x9D, 0x36, 0x42, 0x4A, 0x5E, 0xC1, 0xE0
        },
/*   p1:    */
/*   dpMax       = 10.   lpMax       = 64.   cycleCnt=    2   0   0   1.            */
/* 28BDF76E31940AC5.1E2B4C376DA5F908.4C75169A0ED82B3F.B951C3DE647F208A. */
/* Karnaugh maps:
*   0011 1001 0010 0111. 1010 0111 0100 0110. 0011 0001 1111 0100. 1111 1000 0001 1100.
```

```
*   1100 1111 1111 1010. 0011 0011 1110 0100. 1001 0110 0100 0011. 0101 0110 1011 1011.
*   0010 0100 0011 0101. 1100 1000 1000 1110. 0111 1111 0010 0110. 0000 1010 0000 0011.
*   1101 1000 0010 0001. 0110 1001 1110 0101. 0001 0100 0101 0111. 0011 1011 1111 0010.
*/
        {
        0x75, 0xF3, 0xC6, 0xF4, 0xDB, 0x7B, 0xFB, 0xC8,
        0x4A, 0xD3, 0xE6, 0x6B, 0x45, 0x7D, 0xE8, 0x4B,
        0xD6, 0x32, 0xD8, 0xFD, 0x37, 0x71, 0xF1, 0xE1,
        0x30, 0x0F, 0xF8, 0x1B, 0x87, 0xFA, 0x06, 0x3F,
        0x5E, 0xBA, 0xAE, 0x5B, 0x8A, 0x00, 0xBC, 0x9D,
        0x6D, 0xC1, 0xB1, 0x0E, 0x80, 0x5D, 0xD2, 0xD5,
        0xA0, 0x84, 0x07, 0x14, 0xB5, 0x90, 0x2C, 0xA3,
        0xB2, 0x73, 0x4C, 0x54, 0x92, 0x74, 0x36, 0x51,
        0x38, 0xB0, 0xBD, 0x5A, 0xFC, 0x60, 0x62, 0x96,
        0x6C, 0x42, 0xF7, 0x10, 0x7C, 0x28, 0x27, 0x8C,
        0x13, 0x95, 0x9C, 0xC7, 0x24, 0x46, 0x3B, 0x70,
        0xCA, 0xE3, 0x85, 0xCB, 0x11, 0xD0, 0x93, 0xB8,
        0xA6, 0x83, 0x20, 0xFF, 0x9F, 0x77, 0xC3, 0xCC,
        0x03, 0x6F, 0x08, 0xBF, 0x40, 0xE7, 0x2B, 0xE2,
        0x79, 0x0C, 0xAA, 0x82, 0x41, 0x3A, 0xEA, 0xB9,
        0xE4, 0x9A, 0xA4, 0x97, 0x7E, 0xDA, 0x7A, 0x17,
        0x66, 0x94, 0xA1, 0x1D, 0x3D, 0xF0, 0xDE, 0xB3,
        0x0B, 0x72, 0xA7, 0x1C, 0xEF, 0xD1, 0x53, 0x3E,
        0x8F, 0x33, 0x26, 0x5F, 0xEC, 0x76, 0x2A, 0x49,
        0x81, 0x88, 0xEE, 0x21, 0xC4, 0x1A, 0xEB, 0xD9,
        0xC5, 0x39, 0x99, 0xCD, 0xAD, 0x31, 0x8B, 0x01,
        0x18, 0x23, 0xDD, 0x1F, 0x4E, 0x2D, 0xF9, 0x48,
        0x4F, 0xF2, 0x65, 0x8E, 0x78, 0x5C, 0x58, 0x19,
        0x8D, 0xE5, 0x98, 0x57, 0x67, 0x7F, 0x05, 0x64,
        0xAF, 0x63, 0xB6, 0xFE, 0xF5, 0xB7, 0x3C, 0xA5,
        0xCE, 0xE9, 0x68, 0x44, 0xE0, 0x4D, 0x43, 0x69,
        0x29, 0x2E, 0xAC, 0x15, 0x59, 0xA8, 0x0A, 0x9E,
        0x6E, 0x47, 0xDF, 0x34, 0x35, 0x6A, 0xCF, 0xDC,
        0x22, 0xC9, 0xC0, 0x9B, 0x89, 0xD4, 0xED, 0xAB,
        0x12, 0xA2, 0x0D, 0x52, 0xBB, 0x02, 0x2F, 0xA9,
        0xD7, 0x61, 0x1E, 0xB4, 0x50, 0x04, 0xF6, 0xC2,
        0x16, 0x25, 0x86, 0x56, 0x55, 0x09, 0xBE, 0x91
        }
        };


        ////////////////////////////////////////////////////////////////////////////
        /*****************************************************************************
        TWOFISH2.C    -- Optimized C API calls for TWOFISH AES submission

        Submitters:
                Bruce Schneier, Counterpane Systems
                Doug Whiting,   Hi/fn
                John Kelsey,    Counterpane Systems
                Chris Hall,             Counterpane Systems
                David Wagner,   UC Berkeley
```

```
        Code Author:            Doug Whiting,    Hi/fn

        Version   1.00          April 1998

        Copyright 1998, Hi/fn and Counterpane Systems.   All rights reserved.

        Notes:
                *       Optimized version
                *       Tab size is set to 4 characters in this file

*****************************************************************************/
/*
#include "stdafx.h"

#include"aes.h"
#include"table.h"

#include<memory.h>
#include<assert.h>
*/
//#define GetCodeSize //Uyama



/*
+*****************************************************************************
*                               Constants/Macros/Tables
-*****************************************************************************/

//#define       CONST                                   /* help syntax from C++, NOP here */

//              fullSbox = MDStab;              /* not actually const.   Initialized ONE time */
int                     needToBuildMDS1 = 1;            /* is MDStab initialized yet? */

                keyInstance key = new keyInstance();

/* number of rounds for various key sizes:   128, 192, 256 */
/* (ignored for now in optimized code!) */
int[]   numRounds= {0,ROUNDS_128,ROUNDS_192,ROUNDS_256};

//#if REENTRANT
int[][]   _sBox_  =   key.sBox8x32;
//#endif
//int _sBox8_(int N){return (((byte[]) _sBox_) + (N)*256);}

/*------- see what level of S-box precomputation we need to do -----*/

//#else   /* default is FULL_KEY */
```

```
int                 FULL_KEY =      1;

int TAB_STR = 1;

String    MOD_STRING ="(Full keying)";// TAB_STR

/* Fe32_ does a full S-box + MDS lookup.    Need to #define _sBox_ before use.
    Note that we "interleave" 0,1, and 2,3 to avoid cache bank collisions
    in optimized assembly language.
*/
/*
void      Fe32_(int x, int R){ (key.sBox8x32[0][2*_b(x,R    )] ^ key.sBox8x32[0][2*_b(x,R+1)+1] ^
                    key.sBox8x32[2][2*_b(x,R+2)] ^ key.sBox8x32[2][2*_b(x,R+3)+1]);
}
*/
                    /* set a single S-box value, given the input byte */
void sbSet(int N, int i, int J, int v) { key.sBox8x32[N&2][2*i+(N&1)+2*J]=MDStab[N][v]; }
int       GetSboxKey = 1;


String moduleDescription="Optimized C ";
String modeString                   =MOD_STRING;


/* macro(s) for debugging help */
int                 CHECK_TABLE =       0;              /*  nonzero --> compare against "slow"
table */
int                 VALIDATE_PARMS =    0;              /* disable for full speed */

//#include          "debug.h"                           /* debug display macros */

/* end of debug macros */
/*
#ifdef GetCodeSize
extern int Here(int x);                     // return caller's address!
int TwofishCodeStart() { return Here(0); }
#endif
*/

/*
+*****************************************************************************
*
* Function Name:          TableOp
*
* Function:                   Handle table use checking
*
* Arguments:              op      =       what to do      (see TAB_* defns in AES.H)
*
* Return:                     TRUE --> done (for TAB_QUERY)
*
* Notes: This routine is for use in generating the tables KAT file.
```

```
*                   For this optimized version, we don't actually track table usage,
*                   since it would make the macros incredibly ugly.    Instead we just
*                   run for a fixed number of queries and then say we're done.
*
-***************************************************************************/
int TableOp(int op)
        {
        int queryCnt=0;

        switch (op)
                {
                case 0://TAB_DISABLE:
                        break;
                case 1://TAB_ENABLE:
                        break;
                case 2://TAB_RESET:
                        queryCnt=0;
                        break;
                case 3://TAB_QUERY:
                        queryCnt++;
                        if (queryCnt < TAB_MIN_QUERY)
                                return FALSE;
                }
        return TRUE;
        }


/*
+***************************************************************************
*
* Function Name:        ParseHexDword
*
* Function:                     Parse ASCII hex nibbles and fill in key/iv dwords
*
* Arguments:            bit                     =       # bits to read
*                                       srcTxt          =       ASCII source
*                                       d                       =       ptr to dwords to fill in
*                                       dstTxt          =       where to make a copy of ASCII
source
*                                                                                       (NULL ok)
*
* Return:                       Zero if no error.    Nonzero --> invalid hex or length
*
* Notes:    Note that the parameter d is a DWORD array, not a byte array.
*           This routine is coded to work both for little-endian and big-endian
*           architectures.    The character stream is interpreted as a LITTLE-ENDIAN
*           byte stream, since that is how the Pentium works, but the conversion
*           happens automatically below.
*
-***************************************************************************/
int ParseHexDword(int bits, byte[] srcTxt, int[] d, byte[] dstTxt)
```

```
        {
        int i;
        char c;
        int b;

        class clv /* make sure LittleEndian is defined correctly */
                {
                public byte[]   b = new byte[4];
                public int[] d = new int[1];
                }
        clv v = new clv();

        v.d[0]=1;
        if (v.b[0 ^ ADDR_XOR] != 1)
                return BAD_ENDIAN;              /* make sure compile-time switch is set ok */

        for (i=0;i*32<bits;i++)
                d[i]=0;                                         /* first, zero the field */

        for (i=0;i*4<bits;i++)                  /* parse one nibble at a time */
                {                                                       /* case out the hexadecimal
characters */
                c=(char) srcTxt[i];
                if (dstTxt!=null) dstTxt[i]=(byte) c;
                if ((c >= '0') && (c <= '9'))
                        b=c-'0';
                else if ((c >= 'a') && (c <= 'f'))
                        b=c-'a'+10;
                else if ((c >= 'A') && (c <= 'F'))
                        b=c-'A'+10;
                else
                        return BAD_KEY_MAT;  /* invalid hex character */
                /* works for big and little endian! */
                d[i/8] |= b << (4*((i^1)&7));
                }

        return 0;                                               /* no error */
        }


/*
+*****************************************************************************
*
* Function Name:        RS_MDS_encode
*
* Function:                     Use (12,8) Reed-Solomon code over GF(256) to produce
*                                       a key S-box dword from two key material dwords.
*
* Arguments:            k0      =       1st dword
*                                       k1      =       2nd dword
*
```

```
* Return:                      Remainder polynomial generated using RS code
*
* Notes:
*       Since this computation is done only once per reKey per 64 bits of key,
*       the performance impact of this routine is imperceptible. The RS code
*       chosen has "simple" coefficients to allow smartcard/hardware implementation
*       without lookup tables.
*
-*************************************************************************/
int RS_MDS_Encode(int k0, int k1)
        {
        int i,j;
        int r;

        for (i=r=0;i<2;i++)
                {
                if(0!=i){ r = k0;}
                else{r = k1;}
                for (j=0;j<4;j++)                       /* shift one byte at a time */
                        RS_rem(r);
                }
        return r;
        }


/*
+*************************************************************************
*
* Function Name:        BuildMDS
*
* Function:                     Initialize the MDStab array
*
* Arguments:            None.
*
* Return:                       None.
*
* Notes:
*       Here we precompute all the fixed MDS table.  This only needs to be done
*       one time at initialization, after which the table is "CONST".
*
-*************************************************************************/
int     _b(int[] x, int N){return    (x[((N) & 3)] ^ ADDR_XOR) ;}/* pick bytes out of a dword */

int     b0(int[] x){return                      _b(x,0);} /* extract LSB of DWORD */
int     b1(int[] x){return                      _b(x,1);}
int     b2(int[] x){return                      _b(x,2);}
int     b3(int[] x){return                      _b(x,3);}       /* extract MSB of DWORD */

void SetMDS(int N,int i, int d){
        d = (byte)(M00(P8x8[N][0]));
        d |= (byte)(M00(P8x8[N][0]>>>8));
```

```
        d |= (byte)(M22(P8x8[N][0]>>>16));
        d |= (byte)(M33(P8x8[N][0]>>>24));
        MDStab[N][i] = d;
}

void BuildMDS()
        {
        int i;
        int d = 0;
        int[] m1 = new int[2];
        int[] mX = new int[2];
        int[] mY = new int[4];

        for (i=0;i<256;i++)
                {
                m1[0]= P8x8[0][i];              /* compute all the matrix elements */
                mX[0]= Mul_X(m1[0]);
                mY[0]= Mul_Y(m1[0]);

                m1[1]= P8x8[1][i];
                mX[1]= Mul_X(m1[1]);
                mY[1]= Mul_Y(m1[1]);
/*
#undef  Mul_1                                   // change what the pre-processor does with Mij
#undef  Mul_X
#undef  Mul_Y
#define Mul_1   m1                              // It will now access m01[], m5B[], and mEF[]
#define Mul_X   mX
#define Mul_Y   mY
*/

                SetMDS(0,i,d);                          /* fill in the matrix with elements
computed above */
                SetMDS(1,i,d);
                SetMDS(2,i,d);
                SetMDS(3,i,d);
                }
/*
        #undef  Mul_1
#undef  Mul_X
#undef  Mul_Y
#define Mul_1   Mx_1                    // re-enable true multiply
#define Mul_X   Mx_X
#define Mul_Y   Mx_Y
        */

//      needToBuildMDS=0;                       /* NEVER modify the table again! */
        }

/*
+*****************************************************************************
```

```
*
* Function Name:        ReverseRoundSubkeys
*
* Function:             Reverse order of round subkeys to switch between encrypt/decrypt
*
* Arguments:            key             =       ptr to keyInstance to be reversed
*                       newDir  =       new direction value
*
* Return:               None.
*
* Notes:
*       This optimization allows both blockEncrypt and blockDecrypt to use the same
*       "fallthru" switch statement based on the number of rounds.
*       Note that key->numRounds must be even and >= 2 here.
*
-*****************************************************************************/
void ReverseRoundSubkeys(keyInstance key, int dIR_ENCRYPT2)
        {
        int t0,t1;
        int[] r0=key.subKeys;//+ROUND_SUBKEYS;
        int[] r1=r0;// + 2*key.numRounds - 2;

        int ir0 = ROUND_SUBKEYS;
        int ir1 = ROUND_SUBKEYS + 2*key.numRounds - 2;
        for (;ir0 < ir1;ir0+=2,ir1-=2)
                {
                t0=r0[0+ROUND_SUBKEYS];                 /* swap the order */
                t1=r0[1+ROUND_SUBKEYS];
                r0[0]=r1[0+ 2*key.numRounds - 2];       /* but keep relative order within
pairs */
                r0[1]=r1[1+ 2*key.numRounds - 2];
                r1[0]=t0;
                r1[1]=t1;
                }

        key.direction=dIR_ENCRYPT2;
        }

/*
+*****************************************************************************
*
* Function Name:        Xor256
*
* Function:             Copy an 8-bit permutation (256 bytes), xoring with a byte
*
* Arguments:            dst             =       where to put result
*                       src             =       where to get data (can be same
asa dst)
*                       b               =       byte to xor
*
* Return:               None
```

```
*
* Notes:
*       BorlandC's optimization is terrible!    When we put the code inline,
*       it generates fairly good code in the *following* segment (not in the Xor256
*       code itself).    If the call is made, the code following the call is awful!
*       The penalty is nearly 50%!    So we take the code size hit for inlining for
*       Borland, while Microsoft happily works with a call.
*
-************************************************************************/
                                        /* do it as a function call */
void X_8(int N, int x, int[] d, int[] s)        { d[N]=s[N] ^ x; d[N+1]=s[N+1] ^ x; }
void X_32(int N, int x, int[] d, int[] s)       {       X_8(N,x,d,s);       X_8(N+2,x,d,s);       X_8(N+4,x,d,s);
X_8(N+6,x,d,s); }

void Xor256(int[] dst, int[] src, byte b)
        {
        int         x=b*0x01010101;         /* replicate byte to all four bytes */
        int[] d=(int[])dst;
        int[] s=(int[])src;

        X_32(0,x,d,s ); X_32( 8,x,d,s); X_32(16,x,d,s); X_32(24,x,d,s);  /* all inline */
//      d+=32;   /* keep offsets small! */
//      s+=32;
        X_32(0 ,x,d,s); X_32( 8,x,d,s); X_32(16,x,d,s); X_32(24,x,d,s);  /* all inline */
        }


/*
+************************************************************************
*
* Function Name:        reKey
*
* Function:                     Initialize the Twofish key schedule from key32
*
* Arguments:            key                     =       ptr to keyInstance to be initialized
*
* Return:                       TRUE on success
*
* Notes:
*       Here we precompute all the round subkeys, although that is not actually
*       required.    For example, on a smartcard, the round subkeys can
*       be generated on-the-fly     using f32()
*
-************************************************************************/
int b0(int k){
        int j = 0;
        j = k & 0x000000ff;
        return j;
}
int b1(int k){
        int j = 0;
```

```
        j = (k & 0x0000ff00)>>>8;
        return j;
}
int b2(int k){
        int j = 0;
        j = (k & 0x00ff0000)>>>16;
        return j;
}
int b3(int k){
        int j = 0;
        j = (k & 0xff000000)>>>24;
        return j;
}


void    F32(int res, int x, int[] k32)
{

int t=x;
int ts;
switch (k64Cnt & 3)

    {

        case 0:  /* same as 4 */
                                ts    = (P8x8[P_04][b0(t)] ^ b0(k32[3]))&0x000000ff;

                                ts    |= (P8x8[P_14][b1(t)] ^ b1(k32[3]))<<8;
                                ts    |= (P8x8[P_24][b2(t)] ^ b2(k32[3]))<<16;
                                ts    |= (P8x8[P_34][b3(t)] ^ b3(k32[3]))<<24;
                                t = ts;
                        /* fall thru, having pre-processed t */
        case 3:             ts    = (P8x8[P_03][b0(t)] ^ b0(k32[2]))&0x000000ff;
                                ts    |= (P8x8[P_13][b1(t)] ^ b1(k32[2]))<<8;
                                ts    |= (P8x8[P_23][b2(t)] ^ b2(k32[2]))<<16;
                                ts    |= (P8x8[P_33][b3(t)] ^ b3(k32[2]))<<24;
                                t = ts;
                        /* fall thru, having pre-processed t */
        case 2:   /* 128-bit keys (optimize for this case) */
                res=    MDStab[0][P8x8[P_01][P8x8[P_02][b0(t)]  ^  b0(k32[1])]  ^  b0(k32[0])]  ^

                                MDStab[1][P8x8[P_11][P8x8[P_12][b1(t)]    ^    b1(k32[1])]    ^
b1(k32[0])] ^
                                MDStab[2][P8x8[P_21][P8x8[P_22][b2(t)]    ^    b2(k32[1])]    ^
b2(k32[0])] ^
                                MDStab[3][P8x8[P_31][P8x8[P_32][b3(t)]    ^    b3(k32[1])]    ^
b3(k32[0])] ;
        }

}
```

```
////////////////////////////////////////////////////
byte b_N(int x){return (byte)x;}

void      one128(int N, int i, int J, int N_1, int[] L0, int k0){   sbSet(N,i,J,P8x8[N_1][L0[i+J]]^k0);}
void      sb128(int N, int N_2, int[] L0, int k0, int[] sKey) {
          Xor256(L0,P8x8[N_2],b_N(sKey[1]));
          { k0=b_N(sKey[0]);
          for (i=0;i<256;i+=2) { one128(N,0, N_2, k0, L0, sKey[0]); one128(N,1, N_2, k0, L0, sKey[0]); } } }

void one192(int N, int J, int N_1, int N_2, int[] L0, int k1, int
k0){ sbSet(N,i,J,P8x8[N_1][P8x8[N_2][L0[i+J]]^k1]^k0);}
void      sb192(int N, int N_1, int N_3, int[] L0, int k1, int k0, int[]sKey) {

          Xor256(L0,P8x8[N_3], b_N(sKey[2]))          ;
          { k0=b_N(sKey[0]);
            k1=b_N(sKey[1]);
            for (i=0;i<256;i+=2) { one192(N,0, N_1, N_3, L0, k1, sKey[0]); one192(N,1, N_1, N_3, L0, k1,
sKey[0]); } } }

void one256(int N,int i, int J, int N_1, int N_2, int[] L0, int k1, int
k0){ sbSet(N,i,J,P8x8[N_1][P8x8[N_2][L0[i+J]]^k1]^k0);}
void      sb256(int N, int N_3, int N_4,int[] L0, int[] L1, int k1, int k0, int[]sKey) {

          Xor256(L1,P8x8[N_4],b_N(sKey[3]));
          for (i=0;i<256;i+=2) {L0[i    ]=P8x8[N_3][L1[i]];
                                                    L0[i+1]=P8x8[N_3][L1[i+1]]; }
          Xor256(L0,L0,b_N(sKey[2]));
          { k0=b_N(sKey[0]);
            k1=b_N(sKey[1]);
            for (i=0;i<256;i+=2) { one256(N,0, N_3, N_4,L1[0], L0, k1, k0); one256(N,1, N_3, N_4, L1[0],
L0, k1, k0); } } }

////////////////////////////////////////////////////////////////////
int                 i,j,k64Cnt,keyLen;
int[][] MDStab = new int[4][256];
int reKey(keyInstance key)
          {
//        int                 i,j,k64Cnt,keyLen;
          int                 subkeyCnt=0;
          int       A=0,B=0,q;
          int[]     sKey = new int[MAX_KEY_BITS/64];
          int[] k32e = new int[MAX_KEY_BITS/64];
          int[] k32o = new int[MAX_KEY_BITS/64];
          byte[]    L0 = new byte[256];
          byte[] L1 = new byte[256];          /* small local 8-bit permutations */


          if (needToBuildMDS1 != 0)                    /* do this one time only */
```

```
        BuildMDS();

    if (0==(useAsm & 4))
    {
        subkeyCnt = ROUND_SUBKEYS + 2*key.numRounds;
        keyLen=key.keyLen;
        k64Cnt=(keyLen+63)/64;                  /* number of 64-bit key words */
        for (i=0,j=k64Cnt-1;i<k64Cnt;i++,j--)
        {                                                       /* split into even/odd key dwords */
            k32e[i]=key.key32[2*i  ];
            k32o[i]=key.key32[2*i+1];
            /* compute S-box keys using (12,8) Reed-Solomon code over GF(256) */
            sKey[j]=key.sboxKeys[j]=RS_MDS_Encode(k32e[i],k32o[i]);  /* reverse order */
        }
    }

    for (i=q=0;i<subkeyCnt/2;i++,q+=SK_STEP)
        {                                                       /* compute round subkeys for PHT */
        F32(A,q         ,k32e);         /* A uses even key dwords */
        F32(B,q+SK_BUMP,k32o);                  /* B uses odd  key dwords */
        B = ROL(B,8);
        key.subKeys[2*i  ] = A+B;       /* combine with a PHT */
        B = A + 2*B;
        key.subKeys[2*i+1] = ROL(B,SK_ROTL);
        }

    switch (keyLen)  /* case out key length for speed in generating S-boxes */
        {
        case 128:
//              sb128(0); sb128(1); sb128(2); sb128(3);
            break;
        case 192:
//              sb192(0); sb192(1); sb192(2); sb192(3);
            break;
        case 256:
//              sb256(0); sb256(1);     sb256(2); sb256(3);
            break;
        }

//  DebugDumpKey(key);

    if (key.direction == DIR_ENCRYPT)
        ReverseRoundSubkeys(key,DIR_ENCRYPT);       /* reverse the round subkey order */

    return TRUE;
    }
/*
```

```
+*****************************************************************************
*
* Function Name:        makeKey
*
* Function:                     Initialize the Twofish key schedule
*
* Arguments:    key                     =       ptr to keyInstance to be initialized
*                                       direction       =       DIR_ENCRYPT             or
DIR_DECRYPT
*                                       keyLen          =       # bits of key text at *keyMaterial
*                                       keyMaterial     =       ptr     to      hex     ASCII   chars
representing key bits
*
* Return:                       TRUE on success
*                                       else error code (e.g., BAD_KEY_DIR)
*
* Notes: This parses the key bits from keyMaterial.   Zeroes out unused key bits
*
-*****************************************************************************/
int makeKey(keyInstance key, int dIR_ENCRYPT2, int keyLen, byte[] keyMaterial)
        {


        key.direction   = dIR_ENCRYPT2;/* set our cipher direction */
        key.keyLen              = (keyLen+63) & ~63;            /* round up to multiple of 64 */
        key.numRounds  = numRounds[(keyLen-1)/64];
//      memset(key.key32,0,(key.key32.length));    /* zero unused bits */
        for(i =0; i<key.key32.length; i++){
                key.key32[i] = 0;
        }
        key.keyMaterial[MAX_KEY_SIZE]=0;        /* terminate ASCII string */

        if ((keyMaterial == null) || (keyMaterial[0]==0))
                return TRUE;                    /* allow a "dummy" call */

        if (0!=ParseHexDword(keyLen,keyMaterial,key.key32,key.keyMaterial))
                return BAD_KEY_MAT;

        return reKey(key);                      /* generate round subkeys */
        }


/*
+*****************************************************************************
*
* Function Name:        cipherInit
*
* Function:                     Initialize the Twofish cipher in a given mode
*
* Arguments:    cipher          =       ptr to cipherInstance to be initialized
*                                       mode            =       MODE_ECB,   MODE_CBC,   or
```

```
MODE_CFB1
*                                       IV                      =       ptr to hex ASCII test
representing IV bytes
*
* Return:                       TRUE on success
*                                       else error code (e.g., BAD_CIPHER_MODE)
*
-*****************************************************************************/
int cipherInit(cipherInstance cipher, int mode, byte[] IV)
        {
        int i;

        if ((mode != MODE_ECB) && (IV !=null))   /* parse the IV */
                {
                if (ParseHexDword(BLOCK_SIZE,IV,cipher.iv32,null) !=0)
                        return BAD_IV_MAT;
                for (i=0;i<BLOCK_SIZE/32;i++)    /* make byte-oriented copy for CFB1 */
                        (cipher.IV)[i] = (byte) Bswap(cipher.iv32[i]);
                }

        cipher.mode             =       mode;

        return TRUE;
        }

/*
+*****************************************************************************
*
* Function Name:        blockEncrypt
*
* Function:                     Encrypt block(s) of data using Twofish
*
* Arguments:            cipher          =       ptr to already initialized cipherInstance
*                                       key                     =       ptr to already initialized
keyInstance
*                                       input           =       ptr to data blocks to be
encrypted
*                                       inputLen        =       # bits to encrypt (multiple of
blockSize)
*                                       outBuffer       =       ptr to where to put encrypted
blocks
*
* Return:                       # bits ciphered (>= 0)
*                                       else error code (e.g., BAD_CIPHER_STATE,
BAD_KEY_MATERIAL)
*
* Notes: The only supported block size for ECB/CBC modes is BLOCK_SIZE bits.
*                  If inputLen is not a multiple of BLOCK_SIZE bits in those modes,
*                  an error BAD_INPUT_LEN is returned.   In CFB1 mode, all block
*                  sizes can be supported.
*
```

```
-*****************************************************************************/


byte _b(int x, int N){
        int t = N&3;
        byte bi = (byte)(x>>>8*t);
        return bi;
}

int Fe32_(int x, int R){return (key.sBox8x32[0][2*_b(x,R   )] ^ key.sBox8x32[0][2*_b(x,R+1)+1] ^

                key.sBox8x32[2][2*_b(x,R+2)] ^ key.sBox8x32[2][2*_b(x,R+3)+1]);}

void    LoadBlockE(int N, byte[] ct, byte[] input, int[] sk, int[] IV){   ct[N]=(byte) (Bswap((input)[N])
^ sk[INPUT_WHITEN+N] ^ IV[N]);}


int t0, t1;
int[] x = new int[16];;
void    EncryptRound(int K, int R, int id, int[] sk){
        t0              = Fe32_(x[K   ],0);
        t1              = Fe32_(x[K^1],3);
        x[K^3] = ROL(x[K^3],1);
        x[K^2]^= t0 +     t1 + sk[ROUND_SUBKEYS+2*(R)   ];
        x[K^3]^= t0 + 2*t1 + sk[ROUND_SUBKEYS+2*(R)+1];
        x[K^2] = ROR(x[K^2],1);
//      DebugDump(x,"",rounds-(R),0,0,1,0);
}
void            Encrypt2(int R, int id, int[] sk)     {                       EncryptRound(0,R+1,id,sk);
EncryptRound(2,R,id,sk); }

void    StoreBlockE(int N, byte[] ct, int[] x, int[] sk){       (ct)[N]=(byte)          (x[N^2]         ^
sk[OUTPUT_WHITEN+N]);}


int blockEncrypt(cipherInstance cipher, keyInstance key, byte[] input,
                                int inputLen, byte[] ct)
        {
        int     i,n;                                                    /* loop counters */
        int[]   x = new int[BLOCK_SIZE/32];                             /* block being encrypted */
        byte[] xb = new byte[BLOCK_SIZE/8];
        int t0,t1;                                                      /* temp variables */
        int         rounds=key.numRounds;       /* number of rounds */
        byte   bit,bit0,ctBit,carry;            /* temps for CFB */

        /* make local copies of things for faster access */
        int         mode = cipher.mode;
        int[] sk = new int[TOTAL_SUBKEYS];
        int[] IV = new int[BLOCK_SIZE/32];

//      GetSboxKey;
```

```
        if (mode == MODE_CFB1)
                {       /* use recursion here to handle CFB, one block at a time */
                cipher.mode = MODE_ECB;         /* do encryption in ECB */
                for (n=0;n<inputLen;n++)
                        {
                        blockEncrypt(cipher, key, cipher.IV, BLOCK_SIZE,    xb);
                        bit0    = (byte) (0x80 >>> (n & 7));/* which bit position in byte */
                        ctBit = (byte) ((input[n/8] & bit0) ^ ((( x)[0] & 0x80) >> (n&7)));
                        ct[n/8] = (byte) ((ct[n/8] & ~ bit0) | ctBit);
                        carry = (byte) (ctBit >> (7 - (n&7)));
                        for (i=BLOCK_SIZE/8-1;i>=0;i--)
                                {
                                bit = (byte) (cipher.IV[i] >>> 7);      /* save next "carry" from shift */
                                cipher.IV[i] = (byte) ((cipher.IV[i] << 1) ^ carry);
                                carry = bit;
                                }
                        }
                cipher.mode = MODE_CFB1;        /* restore mode for next time */
                return inputLen;
                }

        /* here for ECB, CBC modes */
        if (key.direction != DIR_ENCRYPT)
                ReverseRoundSubkeys(key,DIR_ENCRYPT);           /* reverse the round subkey
order */


        /* make local copy of subkeys for speed */
//      memcpy(sk,key.subKeys,4*(ROUND_SUBKEYS+2*rounds));
        for(i=0; i<4*(ROUND_SUBKEYS+2*rounds); i++) {
                sk[i] = key.subKeys[i];
        }
        if (mode == MODE_CBC)
                BlockCopy(IV,cipher.iv32);
        else
                IV[0]=IV[1]=IV[2]=IV[3]=0;

//      LoadBlockE(int N, int[] x, int[] input, int[] sk, int[] IV)

        for                                             (n=0;n<inputLen;n++/*+=BLOCK_SIZE*/)//,input,
ct)//+=BLOCK_SIZE/8,ct+=BLOCK_SIZE/8)
                {
                LoadBlockE(0,ct,input,sk,IV);
                LoadBlockE(1,ct,input,sk,IV);
                LoadBlockE(2,ct,input,sk,IV);
                LoadBlockE(3,ct,input,sk,IV);
//              DebugDump(x,"",0,0,0,0,0);
                }
```

```
//          Encrypt2(int R, int id, int[] sk
            /*
                        Encrypt2(14,id,sk);
                        Encrypt2(12,id,sk);
                        Encrypt2(10,_);
                        Encrypt2( 8,_);
                        Encrypt2( 6,_);
                        Encrypt2( 4,_);
                        Encrypt2( 2,_);
                        Encrypt2( 0,_);
*/
                        /* need to do (or undo, depending on your point of view) final swap */
//                      StoreBlockE(int N, int[] outBuffer, int[] x, int[] sk)

                        StoreBlockE(0,ct,x,sk);
                        StoreBlockE(1,ct,x,sk);
                        StoreBlockE(2,ct,x,sk);
                        StoreBlockE(3,ct,x,sk);

                        if (mode == MODE_CBC)
                                {
                                IV[0]=Bswap(ct[0]);
                                IV[1]=Bswap(ct[1]);
                                IV[2]=Bswap(ct[2]);
                                IV[3]=Bswap(ct[3]);
                                }

            if (mode == MODE_CBC){
                        BlockCopy(cipher.iv32,IV);
            }

            return inputLen;
            }

/*
+*****************************************************************************
*
* Function Name:        blockDecrypt
*
* Function:                     Decrypt block(s) of data using Twofish
*
* Arguments:            cipher          =       ptr to already initialized cipherInstance
*                                       key                     =       ptr to already initialized
keyInstance
*                                       input           =       ptr to data blocks to be decrypted
*                                       inputLen        =       # bits to encrypt (multiple of
blockSize)
*                                       outBuffer       =       ptr to where to put decrypted
blocks
*
* Return:                       # bits ciphered (>= 0)
```

```
*                                   else error code (e.g., BAD_CIPHER_STATE,
BAD_KEY_MATERIAL)
*
* Notes: The only supported block size for ECB/CBC modes is BLOCK_SIZE bits.
*                   If inputLen is not a multiple of BLOCK_SIZE bits in those modes,
*                   an error BAD_INPUT_LEN is returned.  In CFB1 mode, all block
*                   sizes can be supported.
*
-*************************************************************************/
void LoadBlockD(int N, byte[] input, int[] sk){ x[N^2]=Bswap((input)[N]) ^
sk[OUTPUT_WHITEN+N];}
void DecryptRound(int K, int R, int id, int[] sk){

                        t0          = Fe32_(x[K  ],0);
                        t1          = Fe32_(x[K^1],3);
//                      DebugDump(x,"",(R)+1,0,0,1,0);
                        x[K^2] = ROL (x[K^2],1);
                        x[K^2]^= t0 +   t1 + sk[ROUND_SUBKEYS+2*(R)  ];
                        x[K^3]^= t0 + 2*t1 + sk[ROUND_SUBKEYS+2*(R)+1];
                        x[K^3] = ROR (x[K^3],1);
}
void Decrypt2(int R, int id, int[] sk)  {  DecryptRound(2,R+1,id,sk);
DecryptRound(0,R,id,sk); }


void StoreBlockD(int N, int[] x, byte[] outBuffer, int[] sk){  (outBuffer)[N] = (byte) (x[N] ^
sk[INPUT_WHITEN+N]);}


void StoreBlockD(int N, int[] x, int[] sk, int[] IV, int[] input, int[] outBuffer){
        x[N]   ^= sk[INPUT_WHITEN+N] ^ IV[N];
IV[N]   = Bswap(((int[])input)[N]);
((int[])outBuffer)[N] = Bswap(x[N]);
}

int blockDecrypt(cipherInstance cipher, keyInstance key, byte[] input,
                                int inputLen, byte[] outBuffer)
        {
        int   i,n;                                          /* loop counters */
        int[] x = new int[BLOCK_SIZE/32];                   /* block being encrypted */
        byte[] xb = new byte[BLOCK_SIZE/8];
        int t0,t1;                                          /* temp variables */
        int         rounds=key.numRounds;       /* number of rounds */
        byte  bit,bit0,ctBit,carry;         /* temps for CFB */

        /* make local copies of things for faster access */
        int         mode = cipher.mode;
        int[]  sk = new int[TOTAL_SUBKEYS];
        int[]  IV = new int[BLOCK_SIZE/32];

//      GetSboxKey;
```

```
        if (cipher.mode == MODE_CFB1)
                {       /* use blockEncrypt here to handle CFB, one block at a time */
                cipher.mode = MODE_ECB;         /* do encryption in ECB */
                for (n=0;n<inputLen;n++)
                        {
                        blockEncrypt(cipher,key,cipher.IV,BLOCK_SIZE, xb);
                        bit0    = (byte) (0x80 >>> (n & 7));
                        ctBit = (byte) (input[n/8] & bit0);
                        outBuffer[n/8] = (byte) ((outBuffer[n/8] & ~ bit0) |
                                                        (ctBit ^ (( x[0] & 0x80) >> (n&7))));
                        carry = (byte) (ctBit >> (7 - (n&7)));
                        for (i=BLOCK_SIZE/8-1;i>=0;i--)
                                {
                                bit = (byte) (cipher.IV[i] >> 7);       /* save next "carry" from shift */
                                cipher.IV[i] = (byte) ((cipher.IV[i] << 1) ^ carry);
                                carry = bit;
                                }
                        }
                cipher.mode = MODE_CFB1;        /* restore mode for next time */
                return inputLen;
                }

        /* here for ECB, CBC modes */
        if (key.direction != DIR_DECRYPT)
                ReverseRoundSubkeys(key,DIR_DECRYPT);           /* reverse the round subkey
order */

        /* make local copy of subkeys for speed */
//      memcpy(sk,key.subKeys,sizeof(DWORD)*(ROUND_SUBKEYS+2*rounds));
        for(i=0; i<4*(ROUND_SUBKEYS+2*rounds); i++) {
                sk[i] = key.subKeys[i];
        }

        if (mode == MODE_CBC)
                BlockCopy(IV,cipher.iv32);
        else
                IV[0]=IV[1]=IV[2]=IV[3]=0;

        for
(n=0;n<inputLen;n++)//BLOCK_SIZE,input+=BLOCK_SIZE/8,outBuffer+=BLOCK_SIZE/8)
                {
//                      DebugDump(input,"¥n",rounds+1,0,0,0,1);
//                      LoadBlockD(int N, int[] input, int[] sk)

                        LoadBlockD(0,input,sk);
                        LoadBlockD(1,input,sk);
                        LoadBlockD(2,input,sk);
                        LoadBlockD(3,input,sk);
```

```
//                      DecryptRound(int K, int R, int id, int[] sk)
                        /*
                        {
                        Decrypt2(14,R,id,sk);
                        Decrypt2(12,_);
                        Decrypt2(10,_);
                        Decrypt2( 8,_);
                        Decrypt2( 6,_);
                        Decrypt2( 4,_);
                        Decrypt2( 2,_);
                        Decrypt2( 0,_);
                        }
*/
//                      DebugDump(x,"",0,0,0,0,0);
                        if (cipher.mode == MODE_ECB)
                                {
//                              StoreBlockD(int N, int[] x, int[] outBuffer, int[] sk)

                                StoreBlockD(0,x,outBuffer,sk);
                                StoreBlockD(1,x,outBuffer,sk);
                                StoreBlockD(2,x,outBuffer,sk);
                                StoreBlockD(3,x,outBuffer,sk);

 // StoreBlockD = 0;
//                              DebugDump(outBuffer,"",-1,0,0,0,1);
                                continue;
                                }
                        else
                                {

                                StoreBlockD(0,x,outBuffer,sk);
                                StoreBlockD(1,x,outBuffer,sk);
                                StoreBlockD(2,x,outBuffer,sk);
                                StoreBlockD(3,x,outBuffer,sk);
//   StoreBlockD = 0;
//                              DebugDump(outBuffer,"",-1,0,0,0,1);
                                }
                        }
            if (mode == MODE_CBC) /* restore iv32 to cipher */
                        BlockCopy(cipher.iv32,IV);

            return inputLen;
            }
/*
#ifdef GetCodeSize
int TwofishCodeSize()
            {
            int x= Here(0);
            return x - TwofishCodeStart();
            };
```

```
#endif
        */

        ////////////////////////////////////////////////////////////////////////////
        // TwofishEC.cpp : コンソール アプリケーションのエントリ ポイントを定義します。
        //
/*
        #include "stdafx.h"

        #include"aes.h"
        #include<stdio.h>
        #include<stdlib.h>
        #include<string.h>
        #include<time.h>
        #include<ctype.h>
*/

//      extern  CONST char *moduleDescription; /* which module is running */
//      extern  CONST char *modeString;                 /* which key schedule mode */
//      extern  CONST int  debugCompile;            /* is external module compiled with
debug? */


        /*
        +*****************************************************************************
        *                       Constants/Macros/Tables
        -*****************************************************************************/

//      typedef struct
        class testData  {
                File f;                         /* the file being written/read */
                int  I;                             /* test number */
                int      keySize;                   /* key size in bits */
                int      gotDebugIO;            /* got any debug IO? */
                byte[]  pt = new byte[BLOCK_SIZE/8];    /* plaintext */
                byte[]  ct = new byte[BLOCK_SIZE/8];    /* ciphertext */

                keyInstance     ki;             /* use ki.keyDwords as key bits */
                cipherInstance ci;              /* use ci.iv as iv bits */
                }


//      char hexTab[]   =       "0123456789ABCDEF";
//      char[]          filePath = new char[128/*80*/];//= "";

//      int                     useAsm          =       0;      /* use assembly language */
//      int                     mctInner        =       MCT_INNER/100;
//      int                     mctOuter        =       MCT_OUTER/10;
//      int                     verify          =       0;      /* set to nonzero to read&verify
files */
//      int                     debug           =       0;      /* debugging mode */
```

```
//      int                     verbose         =       0;      /* verbose output */
//      int                     quietVerify     =       0;      /* quiet during verify */
//      int                     timeIterCnt     =       0;      /* how many times to iterate for
timing */
//      int[]           randBits = new int[64];//= {1};      /* use Knuth's additive generator */
//      int                     randPtr;
//      testData        debugTD                 = null;  /* for use with debugIO */
//      int                     CLKS_BYTE       =       0;      /* use clks/byte? (vs. clks/block)
*/
//      int                     FMT_LOG                 =       0;      /* format for log file */
//      int                     CLK_MHZ                 =       200;/* default clock speed */

//      int             KEY_BITS_0              =       128;                    /* first key bit
setting to test */
//      int             STEP_KEY_BITS           =       ((MAX_KEY_BITS-KEY_BITS_0)/2);

        /*
        static char    hexString[]=

        "0123456789ABCDEFFEDCBA98765432100011223344556677889AABBCCDDEEFF";
        */
//      char[]    hex7String = new char[72];
        //=
        "123456712345671234567123456712345671234567123456712345671234567123456712345671";
        ////////////////////////////////////////////
        /*
        +*****************************************************************************
        *                       Functions
        -*****************************************************************************/
/*
        int Here(int x)
                {
                int mask=~0U;

                return (* (((DWORD *)&x)-1)) & mask;
                }
        extern int TwofishCodeSize(void);
*/

        /*
        +*****************************************************************************
        *
        * Function Name:        Rand
        *
        * Function:                     Generate random number
        *
        * Arguments:            None.
        *
        * Return:                       New random number.
        *
        * Notes:                Uses Knuth's additive generator, other magic
```

```
*
-*************************************************************************/
int Rand()
        {
        if (randPtr >= 57)
                randPtr = 0;                            /* handle the ptr wrap */

        randBits[randPtr] += randBits[(randPtr < 7) ? randPtr-7+57 : randPtr-7];

        randBits[62]+= randBits[61];
        randBits[63] = ROL(randBits[63],9) + 0x6F4ED7D0;        /* very long period! */

        return (randBits[randPtr++] ^ randBits[63]) + randBits[62];
        }


/*
+*************************************************************************
*
* Function Name:        SetRand
*
* Function:                     Initialize random number seed
*
* Arguments:            seed    =       new seed value
*
* Return:                       None.
*
* Notes:
*
-*************************************************************************/
void SetRand(int seed)
        {
        int i;
        int x;

        randPtr=0;
        for (i=x=0;i<64;i++)
                {
                randBits[i]=seed;
                x |= seed;                      /* keep track of lsb of all entries */
                seed = ROL(seed,11) + 0x12345678;
                }

        if ((x & 1) == 0)   /* insure maximal period by having at least one odd value */
                randBits[0]++;

        for (i=0;i<1000;i++)
                Rand();                         /* run it for a while */

        randBits[63] = Rand();
        randBits[62] = Rand();
```

```
        randBits[61] = Rand() | 1;          /* make it odd */
        }


/*
+*****************************************************************************
*
* Function Name:        ClearTestData
*
* Function:                     Initialize test data to all zeroes
*
* Arguments:            t               =       pointer to testData structure
*
* Return:                       None.
*
* Notes:
*
-*****************************************************************************/
void ClearTestData(testData t)
        {
        t.gotDebugIO=0;
//      memset(t.pt,0,BLOCK_SIZE/8);
        for(i=0; i<BLOCK_SIZE/8; i++){t.pt[i] = 0;}
//      memset(t.ct,0,BLOCK_SIZE/8);
        for(i=0; i<BLOCK_SIZE/8; i++){t.ct[i] = 0;}
//      memset(t.ci.iv32,0,BLOCK_SIZE/8);
        for(i=0; i<BLOCK_SIZE/8; i++){t.ci.iv32[i] = 0;}
//      memset(t.ki.key32,0,MAX_KEY_BITS/8);
        for(i=0; i<MAX_KEY_BITS/8; i++){t.ki.key32[i] = 0;}
//      memset(t.ki.keyMaterial,'0',(t.ki.keyMaterial.length));
        for(i=0; i<t.ki.keyMaterial.length; i++){t.ki.keyMaterial[i] = '0';}
        }

/*
+*****************************************************************************
*
* Function Name:        FatalError
*
* Function:                     Output a fatal error message and exit
*
* Arguments:            msg             =       fatal error description (printf string)
*                               msg2    =       2nd parameter to printf msg
*
* Return:                       None.
*
* Notes:
*
-*****************************************************************************/
void FatalError(char[] msg, char[] msg2)
        {
//      printf("¥nFATAL ERROR: ");
```

```
//                      printf(msg,msg2);
//                      exit(2);
                        }


        /*
        +*****************************************************************************
        *
        * Function Name:        AES_FileIO
        *
        * Function:                       Output to file or verify file contents vs. string
        *
        * Arguments:            f               =       opened file
        *                                       s               =       string                  to
output/compare (NULL-->reset, return)
        *                                       errOK   =       do not fatalError on miscompare
        *
        * Return:                         Zero --> compare ok
        *
        * Notes:                  On miscompare, FatalError (unless errOK)
        *
        -*****************************************************************************/
        /*
        int AES_FileIO(File f, byte[] s, int errOK)
                {
                int   i;
                int   lineNum=0;
                int   j=0;
                char[] line = new char[516];//="";

                if (s == null)      // starting new file
                        {
                        line[0]=(char) (j=lineNum=0);
                        return 0;
                        }

                if (0!=verify)
                        {
//                      fprintf(f,s);
                        return 0;
                        }

                // here to verify the file against the string
                for (i=0;s[i]!=0;i++)
                        {
                        while (line[j] == 0)
                                {
                                lineNum++;
                                if (fgets(line,(line.length)-4,f) == null)
                                        {
                                        if ((s[i]=='¥n') && (s[i+1]==0))
```

```
                                        {
                                        line[0]=j=0;        // missing final eol is ok
                                        return 0;
                                        }
//                                      FatalError("Unexpected EOF looking for %s",s);
                                        }
                                if (0!=verbose){
                                //      printf(line);
                                }
                                }
                                j=0;
                                }
                        if (s[i] != line[j])
                                {
                                if ((s[i] == '¥n') && ((i==0) || (s[i-1] == '¥n'))) continue; // blank line
skip

                                if (line[j] == '¥n') {j++; continue;}
                                if (0==errOK)
                                        {
                                        char[] tmp = new char[1024];
//                                      sprintf(tmp,"Miscompare at line #%d:¥n%s¥nlooking
for¥n¥n%%s",lineNum,line);
//                                      FatalError(tmp,s);
                                        }
                                line[0]=(char) (j=0);       // let caller re-synch if desired
                                return 1;                   // return error flag
                                }
                        j++;
                        }

                return 0;
                }
*/



        int atoi( byte s[] ) {
                int i, n, sign;

                for( i = 0; s[i] == ' ' ; i++ )  //先頭の空白を読み飛ばす
                        ;
                sign = ( s[i] == '-' ) ? -1 : 1;      //符号を保存する
                if( s[i] == '-' || s[i] == '+' )      //符号を飛ばす
                        i++;
                for( n = 0; i< s.length - 2 ; i++ )  //s[i]が数字のあいだ、n へ
                        n = 10 * n + ( s[i] - '0' );
                return sign * n;                            //符号を反映
        }
        /*
        +*****************************************************************************
```

```
        *
        * Function Name:        DebugIO
        *
        * Function:                     Output debug string
        *
        * Arguments:            s       =       string to output
        *
        * Return:                       None.
        *
        * Notes:
        *
        -*************************************************************************/
        /*
        void DebugIO(byte[] s)
                {
                if (debugTD!=null)
                        {
                        AES_FileIO(debugTD.f,s,0);
                        debugTD.gotDebugIO=1;
                        }
//              else
//                      printf(s);
                }
*/

        ////////////////////////////////////////////////////////////////
        /*************************************************************/
        ////////////   uyama
        void TwofishEC(String keyfn, String ptfn, String ctfn){
                int numclosed;
                testData t= new testData();;
                int c, block=0;
                int mesLength; //  平文長（バイト）
                byte[] bufp;
                byte[] cstr = new byte[BLOCK_SIZE/8+64];
                // 暗号文格納場所へのポインタ

                File fkey, fin, fout;

            int i, len, rlen=0, blen4, blen;
                int mode,klen, rc=0;
                byte[] c_mode = new byte[3];
                byte[] c_klen = new byte[5];
                int[] c_key = new int[66];
                byte[] c_keyb = null;// new byte[66];
                int[] c_cini = new int[32+2];
                byte[] c_cinib = new byte[32+2];

                blen4 = 2048;

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

```
        fkey = new File(keyfn);
        fkey.getParentFile().mkdir();
        FileInputStream inkeyst=null;
        try {
                inkeyst = new FileInputStream(fkey);
                inkeyst.read( c_mode);
                inkeyst.read( c_klen);
                klen = atoi(c_klen);
                c_keyb = new byte[klen/4+2];
                inkeyst.read( c_keyb );
                inkeyst.read( c_cinib);
        } catch (IOException e5) {
                // TODO 自動生成された catch ブロック
                e5.printStackTrace();
        }

        fin = new File(ptfn);
        fin.getParentFile().mkdir();
        FileInputStream inptst=null;
        try {
                inptst = new FileInputStream(fin);
        } catch (FileNotFoundException e5) {
                // TODO 自動生成された catch ブロック
                e5.printStackTrace();
        }

        fout = new File(ctfn);
        fout.getParentFile().mkdir();
        FileOutputStream outctst=null;
        try {
                outctst = new FileOutputStream(fout);
        } catch (FileNotFoundException e5) {
                // TODO 自動生成された catch ブロック
                e5.printStackTrace();
        }

        mode = atoi(c_mode);
        klen = atoi(c_klen);
        blen = 128;


    /* 鍵*/
        for(i=0; i<klen/4; i++){
                hex7String[i] = c_keyb[i];
        }
        hex7String[klen/4] = 0;


////////////////////////////////////////////////////////////////
    // 平文
```

```
                try{
                int filelen = inptst.available();
                int head = 4;//sizeof(long);
                int s = 4;
                mesLength = filelen + head;

                if (cipherInit(t.ci,mode,hex7String) != TRUE)
//                      FatalError("cipherInit error during %s test",""/*fname*/);
                t.keySize=klen;
                ClearTestData(t);                                       /* start with all zeroes */
                if (makeKey(t.ki,DIR_ENCRYPT,t.keySize,hex7String/*t.ki.keyMaterial*/) != TRUE)
//                      FatalError("Error parsing key during %s test",""/*fname*/);

                //暗号化
                if(mesLength <= BLOCK_SIZE/8){//63 が暗号化の作業サイズ　63*20＝1260
                        bufp = new byte[BLOCK_SIZE/8+64];
                        if(bufp == null){
//                              printf("No memory");
                                return;
                        }
            // 平文
                        bufp[0] = (byte) filelen;
                        bufp[1] = (byte)(filelen>>>8);
                        bufp[2] = (byte)(filelen>>>16);
                        bufp[3] = (byte)(filelen>>>24);

                        if(filelen > blen4-s){
                                len = inptst.read(bufp, 4, blen4-s );
                        }else{
                                len = inptst.read(bufp, 4, filelen);
                        }
                    rlen -= len;

//                      memcpy(t.pt,bufp,BLOCK_SIZE/8);
                        for(i=0; i<BLOCK_SIZE/8;i++){
                                t.pt[i] = bufp[i];
                        }
                  // 暗号化実行
                        if (blockEncrypt(t.ci,t.ki,t.pt,BLOCK_SIZE,t.ct) != BLOCK_SIZE)
//                              FatalError("blockEncrypt return during %s test","ff.bin"/*fname*/);
//                      memcpy(cstr,t.ct,BLOCK_SIZE/8);
                        for(i=0; i<BLOCK_SIZE/8;i++){
                                cstr[i] = t.ct[i];
                        }

                        outctst.write(cstr, 0, BLOCK_SIZE/8);


                }
                else{
                        int rBlen = mesLength;
```

```
                    bufp = new byte[BLOCK_SIZE/8+64];//63 が暗号化の作業サイズ　63*20＝
1260
                    if(bufp == null){
//                          printf("メモリ不足¥r¥n");
                            return;
                    }
                    bufp[0] = (byte) filelen;
                    bufp[1] = (byte)(filelen>>>8);
                    bufp[2] = (byte)(filelen>>>16);
                    bufp[3] = (byte)(filelen>>>24);

                    int r =0;
                do //while(rlen > 0 && finst.available()>0)
                {
                    // read a block and reduce the remaining byte count
                  if(r==0){
                    len = inptst.read(bufp, 4, BLOCK_SIZE/8-head);
                  }else{
                    len = inptst.read(bufp, 0, BLOCK_SIZE/8);
                  }

                  if(rBlen >= BLOCK_SIZE/8){ block = BLOCK_SIZE/8; }//63 が暗号化の作業サイズ
63*20＝1260
                            if(rBlen < BLOCK_SIZE/8){ block = rBlen;}

                  blockEncrypt(t.ci,t.ki,t.pt,BLOCK_SIZE,t.ct);

                 // 暗号文を書き込む
//                          memcpy(cstr,t.ct,BLOCK_SIZE/8);
                            for(i=0; i<BLOCK_SIZE/8; i++){
                                    cstr[i] = t.ct[i];
                            }
                            outctst.write(cstr, 0, BLOCK_SIZE/8);
                            r += 1;
                            rBlen -= block;
                }while(rBlen>0);
                }
                if(outctst != null)
                {
                    try {
                            outctst.close();
                    } catch (IOException e) {
                            // TODO 自動生成された catch ブロック
                            e.printStackTrace();
                    }
                }

                if(inptst != null)
                {
                    try {
                            inptst.close();
```

```
                        } catch (IOException e) {
                                // TODO 自動生成された catch ブロック
                                e.printStackTrace();
                        }
                }
                } catch (IOException e1) {
                        // TODO 自動生成された catch ブロック
                        e1.printStackTrace();
                }
                //numclosed = _fcloseall(); /* 理由は不明だが使えない。 */
//              printf("TfEC End!¥n" );
        }

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

        /*****************************************************************/


        /*
        +*******************************************************************************
        *
        * Function Name:        GiveHelp
        *
        * Function:                     Print out list of command line switches
        *
        * Arguments:            None.
        *
        * Return:                       None.
        *
        * Notes:
        *
        -*******************************************************************************/
        void GiveHelp()
                {
                /*
                printf("Syntax:    TST2FISH [options]¥n"
                        "Purpose:   Generate/validate AES Twofish code and files¥n"
                        "Options:   -lNN      ==> set sanity check loop to NN¥n"
                        "           -m        ==> do full MCT generation¥n"
                        "           -pPath    ==> set file path¥n"
                        "           -s        ==> set initial random seed based on time¥n"
                        "           -sNN      ==> set initial random seed to NN¥n"
                        "           -tNN      ==> time performance using NN iterations¥n"
                        "           -v        ==> validate files, don't generate them¥n",
                        MAX_ROUNDS
                       );

                exit(1);*/
                }
```

```
/*
	void ShowHex(File f, void[] p,int bCnt, byte[] name)
		{
		int i;

		fprintf(f,"	;%s:",name);
		for (i=0;i<bCnt;i++)
			{
			if ((i % 8) == 0)
				fprintf(f,"¥n¥t.byte¥t");
			else
				fprintf(f,",");
//			fprintf(f,"0%02Xh",((BYTE *)p)[i]);
			}
		fprintf(f,"¥n");
		}
*/
	/* output a formatted 6805 test vector include file */
	/*
	void Debug6805()
		{
		int i,j;
		testData t;
		FILE *f;

		ClearTestData(&t);
		t.keySize=128;

		f=stdout;
		cipherInit(&t.ci,MODE_ECB,NULL);
		makeKey(&t.ki,DIR_ENCRYPT,t.keySize,t.ki.keyMaterial);

		for (i=0;i<4;i++)  // make sure it all fits in 256 bytes
			{
			reKey(&t.ki);
			blockEncrypt(&t.ci,&t.ki,t.pt,BLOCK_SIZE,t.ct);
			fprintf(f,"; Twofish vector #%d¥n",i+1);
			ShowHex(f,&t.keySize,1,"Key Size");
			ShowHex(f,t.ki.key32,16,"Key");
			ShowHex(f,t.pt,BLOCK_SIZE/8,"Plaintext");
			ShowHex(f,t.ct,BLOCK_SIZE/8,"Ciphertext");
			for (j=0;j<16;j++)
				((BYTE *)t.ki.key32)[j] = t.pt[j] ^ t.ct[j];
			memcpy(t.pt,t.ct,sizeof(t.pt));
			fprintf(f,";-------------------------------------------------------¥n");
			}
		fprintf(f,"¥n¥t.byte 0¥t;end of list¥n");
		fclose(f);
		}
*/
```

```
	//////////////////////////////////////////////
	/*
	int main(int argc, char* argv[])
	{
	int	MAX_ARGS	=40;
		int i,testCnt=32;
		DWORD randSeed=0x12345678;
		String moduleName=moduleDescription;

		// 引数チェック
		if( argc != 4 ){				// 使い方の誤り
			exit(1);
		}

		i=1;				// make sure LittleEndian is defined correctly
		if (b0(i) != 1)
			FatalError("LittleEndian defined incorrectly","");
//		if ((ALIGN32) && (k == 2))
//			FatalError("Cannot enable ALIGN32 in 16-bit mode¥n","");


		SetRand(randSeed);					// init pseudorandom
generator for testing



		UY_EC_TF(argv[1],argv[2],argv[3]);

		return 0;
		}

*/



	//////////////////////////////////////////////////////////////////////////////////////////
	// TwofishDC.cpp : コンソール アプリケーションのエントリ ポイントを定義します。
	//
/*
	#include "stdafx.h"

	#include"aes.h"
	#include<stdio.h>
	#include<stdlib.h>
	#include<string.h>
	#include<time.h>
	#include<ctype.h>
*/
//	extern	CONST char *moduleDescription; /* which module is running */
//	extern	CONST char *modeString;				/* which key schedule mode */
//	extern	CONST int	debugCompile;			/* is external module compiled with
```

```
debug? */

        /*
        +*****************************************************************************
        *                               Constants/Macros/Tables
        -*****************************************************************************/

//      typedef struct

//      class testData  {
//              File f;                         /* the file being written/read */
//              int  I;                                 /* test number */
//              int      keySize;                       /* key size in bits */
//              int      gotDebugIO;            /* got any debug IO? */
//              byte[] pt = new byte[BLOCK_SIZE/8];     /* plaintext */
//              byte[] ct = new byte[BLOCK_SIZE/8];     /* ciphertext */

//              keyInstance     ki;             /* use ki.keyDwords as key bits */
//              cipherInstance ci;              /* use ci.iv as iv bits */
//              }


        String hexTab   =       "0123456789ABCDEF";
        char[]          filePath = new char[128/*80*/];//= "";

        int                     useAsm          =       0;      /* use assembly language */
        int                     mctInner        =       MCT_INNER/100;
        int                     mctOuter        =       MCT_OUTER/10;
        int                     verify          =       0;      /* set to nonzero to read&verify
files */
        int                     debug           =       0;      /* debugging mode */
        int                     verbose         =       0;      /* verbose output */
        int                     quietVerify     =       0;      /* quiet during verify */
        int                     timeIterCnt     =       0;      /* how many times to iterate for
timing */
        int[]           randBits = new int[64];//= {1};      /* use Knuth's additive generator */
        int                     randPtr;
        testData        debugTD         ;//     = NULL;         /* for use with debugIO */
        int                     CLKS_BYTE       =       0;      /* use clks/byte? (vs. clks/block)
*/
        int                     FMT_LOG                 =       0;      /* format for log file */
        int                     CLK_MHZ                 =       200;/* default clock speed */

        int             KEY_BITS_0      =               128;                    /* first key bit
setting to test */
        int     STEP_KEY_BITS           =       ((MAX_KEY_BITS-KEY_BITS_0)/2);
        /*
        static char   hexString[]=

        "0123456789ABCDEFFEDCBA98765432100011223344556677889AABBCCDDEEFF";
```

```
        */
        byte[]    hex7String = new byte[72];// use 64 byte
        //=
        "12345671234567123456712345671234567123456712345671234567123456712345671";
        ////////////////////////////////////////////
        /*
        +*****************************************************************************
        *                               Functions
        -*****************************************************************************/
/*
        int Here(int x)
                {
                int mask=~0U;

                return (* (((int[] )&x)-1)) & mask;
                }
                */

//      extern int TwofishCodeSize();


        /*
        +*****************************************************************************
        *
        * Function Name:        Rand
        *
        * Function:                     Generate random number
        *
        * Arguments:            None.
        *
        * Return:                       New random number.
        *
        * Notes:                Uses Knuth's additive generator, other magic
        *
        -*****************************************************************************/
        /*
        int Rand()
                {
                if (randPtr >= 57)
                        randPtr = 0;                            // handle the ptr wrap

                randBits[randPtr] += randBits[(randPtr < 7) ? randPtr-7+57 : randPtr-7];

                randBits[62]+= randBits[61];
                randBits[63] = ROL(randBits[63],9) + 0x6F4ED7D0;        // very long period!

                return (randBits[randPtr++] ^ randBits[63]) + randBits[62];
                }
*/

        /*
```

```
+*****************************************************************************
*
* Function Name:        SetRand
*
* Function:                     Initialize random number seed
*
* Arguments:            seed    =       new seed value
*
* Return:                       None.
*
* Notes:
*
-*****************************************************************************/
/*
void SetRand(int seed)
        {
        int i;
        DWORD x;

        randPtr=0;
        for (i=x=0;i<64;i++)
                {
                randBits[i]=seed;
                x |= seed;                      // keep track of lsb of all entries
                seed = ROL(seed,11) + 0x12345678;
                }

        if ((x & 1) == 0)   // insure maximal period by having at least one odd value
                randBits[0]++;

        for (i=0;i<1000;i++)
                Rand();                         // run it for a while

        randBits[63] = Rand();
        randBits[62] = Rand();
        randBits[61] = Rand() | 1;              // make it odd
        }
*/

        /*
        +*****************************************************************************
        *
        * Function Name:        ClearTestData
        *
        * Function:                     Initialize test data to all zeroes
        *
        * Arguments:            t               =       pointer to testData structure
        *
        * Return:                       None.
        *
        * Notes:
```

```
        *
        -*************************************************************************/
/*
        void ClearTestData(testData t)
                {
                t.gotDebugIO=0;
                memset(t.pt,0,BLOCK_SIZE/8);
                memset(t.ct,0,BLOCK_SIZE/8);
                memset(t.ci.iv32,0,BLOCK_SIZE/8);
                memset(t.ki.key32,0,MAX_KEY_BITS/8);
                memset(t.ki.keyMaterial,'0',sizeof(t.ki.keyMaterial));
                }
*/

        /*
        +*************************************************************************
        *
        * Function Name:        FatalError
        *
        * Function:                     Output a fatal error message and exit
        *
        * Arguments:            msg             =       fatal error description (printf string)
        *                                       msg2    =       2nd parameter to printf msg
        *
        * Return:                       None.
        *
        * Notes:
        *
        -*************************************************************************/
        /*
        void FatalError( char[] msg, char[] msg2)
                {
                printf("¥nFATAL ERROR: ");
                printf(msg,msg2);
                exit(2);
                }
*/

        /*
        +*************************************************************************
        *
        * Function Name:        AES_FileIO
        *
        * Function:                     Output to file or verify file contents vs. string
        *
        * Arguments:            f               =       opened file
        *                                       s               =       string          to
output/compare (NULL-->reset, return)
        *                                       errOK   =       do not fatalError on miscompare
        *
        * Return:                       Zero --> compare ok
```

```
        *
        * Notes:                    On miscompare, FatalError (unless errOK)
        *
        -*************************************************************************/
        /*
        int AES_FileIO(File f, char[] s, int errOK)
                {
                int   i;
                int   lineNum=0;
                int   j=0;
                char[] line = new char[516];//="";

                if (s == null)       // starting new file
                        {
                        line[0]=(char) (j=lineNum=0);
                        return 0;
                        }

                if (0!=verify)
                        {
//                      fprintf(f,s);
                        return 0;
                        }

                // here to verify the file against the string
                for (i=0;s[i]!=0;i++)
                        {
                        while (line[j] == 0)
                                {
                                lineNum++;
                                if (fgets(line, (line.length)-4,f) == null)
                                        {
                                        if ((s[i]=='¥n') && (s[i+1]==0))
                                                {
                                                line[0]=(char) (j=0);          // missing final eol is ok
                                                return 0;
                                                }
//                                      FatalError("Unexpected EOF looking for %s",s);
                                        }
                                if (verbose) printf(line);
                                j=0;
                                }
                        if (s[i] != line[j])
                                {
                                if ((s[i] == '¥n') && ((i==0) || (s[i-1] == '¥n'))) continue; /* blank line
skip
                                if (line[j] == '¥n') {j++; continue; }
                                if (!errOK)
                                        {
                                        char[] tmp = new char[1024];
//                                      sprintf(tmp,"Miscompare   at   line   #%d:¥n%s¥nlooking
```

```
for¥n¥n%%s",lineNum,line);
//                                      FatalError(tmp,s);
                                        }
                                line[0]=(char) (j=0);       // let caller re-synch if desired
                                return 1;                   // return error flag
                                }
                        j++;
                        }

                return 0;
                }

*/


        /*
        +*****************************************************************************
        *
        * Function Name:        DebugIO
        *
        * Function:                     Output debug string
        *
        * Arguments:            s       =       string to output
        *
        * Return:                       None.
        *
        * Notes:
        *
        -*****************************************************************************/
        /*
        void DebugIO( char[] s)
                {
                if (debugTD!=null)
                        {
                        AES_FileIO(debugTD.f,s,0);
                        debugTD.gotDebugIO=1;
                        }
//              else
//                      printf(s);
                }
*/

        ////////////////////////////////////////
        //////////////////////////////////////////////////////////////
        /*****************************************************************/
        ////////////   uyama 復号化
        void TwofishDC(String keyfn, String ctfn, String ptfn){
                testData t = new testData();;
                int head;
                int lenp=0;
                int cc, block=0;
```

```
        byte[] bufc;
        byte[] ostr = new byte[BLOCK_SIZE/8+64];
        // 暗号文格納場所へのポインタ

        byte[] c_mode = new byte[3];
        byte[] c_klen = new byte[5];
        int[] c_key = new int[64+2];
        byte[] c_keyb = null;//new byte[64+2];
        int j,k;
    int len, rlen, blen4, pfilelen;
        int i,mode,klen,blen,rc=0;

        blen4 = 2048;

        cipherInstance cipherI = new cipherInstance();
        keyInstance keyI = new keyInstance();
        ////////////////////////////////////////////////////

        try{
        FileOutputStream foutst = openFileOutput(ptfn,MODE_PRIVATE);
        FileInputStream finst = openFileInput(ctfn);

        File fkey = new File(keyfn);
        fkey.getParentFile().mkdir();
        FileInputStream inkeyst=null;
        try {
                inkeyst = new FileInputStream(fkey);
                inkeyst.read(c_mode);
                inkeyst.read(c_klen);
                klen = atoi(c_klen);
                c_keyb = new byte[klen/4+2];
                inkeyst.read(c_keyb);
        } catch (IOException e3) {
                // TODO 自動生成された catch ブロック
                e3.printStackTrace();
        }//127

        mode = atoi(c_mode);
        klen = atoi(c_klen);
        blen = 128;

        /* 鍵*/
        for(i=0; i<klen/4; i++){
                hex7String[i] = c_keyb[i];
        }
        hex7String[klen/4] = 0;


////////////////////////////////////////////////////////////////
    // 暗文
        int filelen = finst.available();
```

```
            rlen = filelen;
                head = 4;//sizeof(long);

                t.keySize=klen;
                if (cipherInit(t.ci,mode,hex7String) != 0){
//                      FatalError("cipherInit error during %s test",""/*fname*/);
                }
                ClearTestData(t);                                       /* start with all zeroes */
                if (makeKey(t.ki,DIR_DECRYPT,t.keySize,hex7String/*t.ki.keyMaterial*/) != 0){
//                      FatalError("Error parsing key during %s test",""/*fname*/);
                }

                //暗号化
                if(filelen <= BLOCK_SIZE/8){//63 が暗号化の作業サイズ　63*20＝1260
                        bufc = new byte[BLOCK_SIZE/8+64];
                        i=0;
                        if(bufc == null){
//                              printf("No memory");
                                return;
                        }

                        // write the bytes of the file
                        if(blen4<=filelen){
                                len = finst.read(bufc, 0, blen4 );
                        }else{
                                len = finst.read(bufc, 0, filelen );
                        }
                        rlen = rlen - len;

//                      memcpy(t.ct,bufc,BLOCK_SIZE/8);
                        for(i=0; i<BLOCK_SIZE/8; i++){
                                t.ct[i] = bufc[i];
                        }
            // 復号化実行
                        if (blockDecrypt(t.ci,t.ki,t.ct,BLOCK_SIZE,t.pt) != BLOCK_SIZE){
                        //      FatalError("blockDecrypt return during %s test","ff.bin"/*fname*/);
                        }
//                      memcpy(ostr,t.pt,BLOCK_SIZE/8);
                        for(i=0; i<BLOCK_SIZE/8; i++){
                                ostr[i] = t.pt[i];
                        }
                        foutst.write(ostr, head, rlen);

                }
                else{
                        int rBlen = filelen;
                        bufc = new byte[BLOCK_SIZE/8+64];//63 が暗号化の作業サイズ　63*20＝
1260
                        if(bufc == null){
//                              printf("メモリ不足¥r¥n");
                                return;
```

```
                }

                int r = 0;
                do //while(rlen > 0 && finst.available()>0)
        {
                i = 0;
                len = finst.read(bufc, 0, BLOCK_SIZE/8);
                if(rBlen >= BLOCK_SIZE/8){ block = BLOCK_SIZE/8; }//63 が暗号化の作業サイズ
63*20＝1260
                        if(rBlen < BLOCK_SIZE/8){ block = rBlen;}
                        for(i=0; i<BLOCK_SIZE/8; i++){
                                t.ct[i] = bufc[i];
                        }
                    // 復号化実行
                        blockDecrypt(t.ci,t.ki,t.ct,BLOCK_SIZE,t.pt);// != BLOCK_SIZE)
                        for(i=0; i<BLOCK_SIZE/8; i++){
                                ostr[i] = t.ct[i];
                        }
                if(r ==0){
                                byte[] tmpch4 = new byte[4];
                                tmpch4[0] = ostr[0];//(byte) pbuf[0];
                                tmpch4[1] = ostr[1];//(byte) (pbuf[0]>>>8);
                                tmpch4[2] = ostr[2];//(byte) (pbuf[0]>>>16);
                                tmpch4[3] = ostr[3];//(byte) (pbuf[0]>>>24);
                                int jj = 0;
                                int tmp = 0;
                                for (int p = 0; p < tmpch4.length; p++) {
                                        tmp = (tmpch4[3-p] & 0xff);
                                        if(tmp < 0){
                                                tmpch4[3-p] ^= 0x80;
                                                tmp = tmpch4[3-p];
                                                tmp += 0x80;
                                        }
                                        jj = (jj << 8) | tmp;
                                }
                                lenp = jj;
                        foutst.write(ostr, 4, BLOCK_SIZE/8-4);
                        lenp -= BLOCK_SIZE/8-4;
                }
                else{
                        if(lenp >= BLOCK_SIZE/8){
                                foutst.write(ostr, 0, BLOCK_SIZE/8);
                                lenp -= BLOCK_SIZE/8;
                        }
                        else{
                                foutst.write(ostr, 0, lenp);
                                lenp -= lenp;
                        }
                }
                r += 1;
                rBlen -= block;
```

```
		}while(rBlen>0);
		}
		if(inkeyst != null)
		{
			try {
				inkeyst.close();
			} catch (IOException e) {
				// TODO 自動生成された catch ブロック
				e.printStackTrace();
			}
		}

		if(foutst != null)
		{
			try {
				foutst.close();
			} catch (IOException e) {
				// TODO 自動生成された catch ブロック
				e.printStackTrace();
			}
		}

		if(finst != null)
		{
			try {
				finst.close();
			} catch (IOException e) {
				// TODO 自動生成された catch ブロック
				e.printStackTrace();
			}
		}
		} catch (IOException e1) {
			// TODO 自動生成された catch ブロック
			e1.printStackTrace();
		}
}

/*************************************************************/


/*
+*****************************************************************************
*
* Function Name:		GiveHelp
*
* Function:				Print out list of command line switches
*
* Arguments:			None.
*
* Return:				None.
*
```

```
        * Notes:
        *
        -*************************************************************************/
        /*
        void GiveHelp()
                {
                printf("Syntax:     TST2FISH [options]¥n"
                        "Purpose:   Generate/validate AES Twofish code and files¥n"
                        "Options:   -lNN      ==> set sanity check loop to NN¥n"
                        "           -m        ==> do full MCT generation¥n"
                        "           -pPath    ==> set file path¥n"
                        "           -s        ==> set initial random seed based on time¥n"
                        "           -sNN      ==> set initial random seed to NN¥n"
                        "           -tNN      ==> time performance using NN iterations¥n"
                        "           -v        ==> validate files, don't generate them¥n",
                        MAX_ROUNDS
                       );
                exit(1);
                }
                */

/*
        void ShowHex(File f, void[] p,int bCnt, char[] name)
                {
                int i;

                fprintf(f,"       ;%s:",name);
                for (i=0;i<bCnt;i++)
                        {
                        if ((i % 8) == 0)
                                fprintf(f,"¥n¥t.byte¥t");
                        else
                                fprintf(f,",");
                        fprintf(f,"0%02Xh",((BYTE *)p)[i]);
                        }
                fprintf(f,"¥n");
                }
*/
        /* output a formatted 6805 test vector include file */
        /*
        void Debug6805()
                {
                int i,j;
                testData t;
                FILE *f;

                ClearTestData(&t);
                t.keySize=128;

                f=stdout;
                cipherInit(&t.ci,MODE_ECB,NULL);
```

```
                makeKey(&t.ki,DIR_ENCRYPT,t.keySize,t.ki.keyMaterial);

                for (i=0;i<4;i++)  // make sure it all fits in 256 bytes
                        {
                        reKey(&t.ki);
                        blockEncrypt(&t.ci,&t.ki,t.pt,BLOCK_SIZE,t.ct);
                        fprintf(f,"; Twofish vector #%d¥n",i+1);
                        ShowHex(f,&t.keySize,1,"Key Size");
                        ShowHex(f,t.ki.key32,16,"Key");
                        ShowHex(f,t.pt,BLOCK_SIZE/8,"Plaintext");
                        ShowHex(f,t.ct,BLOCK_SIZE/8,"Ciphertext");
                        for (j=0;j<16;j++)
                                ((BYTE *)t.ki.key32)[j] = t.pt[j] ^ t.ct[j];
                        memcpy(t.pt,t.ct,sizeof(t.pt));
                        fprintf(f,";-----------------------------------------------------¥n");
                        }
                fprintf(f,"¥n¥t.byte 0¥t;end of list¥n");
                fclose(f);
                }
*/

        ////////////////////////////////////////////
        /*
        int main(int argc, char* argv[])
        {
        int     MAX_ARGS        =40;
                int i,testCnt=32;
                int   randSeed=0x12345678;
                String moduleName=moduleDescription;

                // 引数チェック
                if( argc != 4 ){                                        // 使い方の誤り
                        exit(1);
                }

                i=1;                            // make sure LittleEndian is defined correctly
                if (b0(i) != 1)
                        FatalError("LittleEndian defined incorrectly","");
//              if ((ALIGN32) && (k == 2))
//                      FatalError("Cannot enable ALIGN32 in 16-bit mode¥n","");



                SetRand(randSeed);                                                      //  init  pseudorandom
generator for testing



                UY_DC_TF(argv[1],argv[2],argv[3]);

                return 0;
```

```
                }
*/


}
```