```
package hiptag;

/*
 * Created on 9 juil. 2009, by Pascal Urien
 * Pascal.Urien@gmail.com
 * Telecom Paris Tech, 37-39 rue Dareau, 75014, Paris, France
 */

import  javacard.framework.*;
import  javacard.security.* ;



public class tag extends Applet

{
        final static byte  SELECT       = (byte)  0xA4  ;
        final static byte  INS_I1       = (byte)  0xCA  ;
        final static byte  INS_I2       = (byte)  0x86  ;

        final static short R_T             = (short)0x400 ;
        final static short HIP_T_TRANSFORM = (short)0x402 ;
        final static short F_T             = (short)0x404 ;
        final static short Signature_T     = (short)0x406 ;
        final static short ESP_Transform   = (short)0x408 ;
        final static short ESP_Info        = (short)0x40A ;

        final static int ALIGN = 8;

        final static short len_r2 =(short)20;

        final static byte [] algo1 = { (byte)0x00,(byte)0x01,(byte)0x00,(byte)0x00 };

        final static byte [] ct1 = {  (byte)0x00,(byte)0x00,(byte)0x00,(byte)0x01,
                                      (byte)'T',(byte)'y',(byte)'p',(byte)'e',
                                      (byte)' ',(byte)'0',(byte)'0',(byte)'0',(byte)'1',
                                      (byte)' ',(byte)'k',(byte)'e',(byte)'y' };


        final static byte [] ct2 = {  (byte)0x00,(byte)0x00,(byte)0x00,(byte)0x02,
                                      (byte)'T',(byte)'y',(byte)'p',(byte)'e',
                                      (byte)' ',(byte)'0',(byte)'0',(byte)'0',(byte)'1',
                                      (byte)' ',(byte)'k',(byte)'e',(byte)'y' };


        static MessageDigest sha1=null ;
        static RandomData rnd=null;


        static byte[] DB =null;
        final static short DBSIZE=(short)200;

        final static short off_myHIT = (short)0   ;
        final static short off_rHIT =  (short)16  ;
        final static short off_R1   =  (short)32  ; //32
        final static short off_R2   =  (short)64  ;
        final static short off_kaut =  (short)96  ;
        final static short off_k    =  (short)128 ;
        final static short off_FT   =  (short)160 ;


        final static byte[] HEADER=  {(byte)0x3b,(byte)0x04,(byte)0x40,(byte)0x11,
                                      (byte)0x00,(byte)0x00,(byte)0x00,(byte)0x00};

        static byte[] MyEPCCODE =     {(byte)0x01,(byte)0x23,(byte)0x45,(byte)0x67,
                                       (byte)0x89,(byte)0xab,(byte)0xcd,(byte)0xef,
                                       (byte)0xcd,(byte)0xab };

        public void init()
        { try { sha1=MessageDigest.getInstance(MessageDigest.ALG_SHA,false); }
          catch (CryptoException e){sha1=null;}

          try { rnd = RandomData.getInstance(RandomData.ALG_SECURE_RANDOM);}
          catch (CryptoException e){}

          DB = JCSystem.makeTransientByteArray(DBSIZE,JCSystem.CLEAR_ON_DESELECT);
        }
```

```
public short GetAttOffset(byte[] pkt, short off, short len, short att)
{ boolean more=true;
  short type=(short)0;
  short tl=(short)0;

  if (len <= (short)40)
         return (short)-1 ;

  while (more)
  { type = Util.getShort(pkt,off)           ;
       tl   = Util.getShort(pkt,(short)(off+2));

       if (type == att) return off    ;

       off =(short)(off+tl) ;

       if (off >= (short)(off+len))
              more=false;

       }

   return -1;
}


public static short GetPadLength(short size)
{ if ( (short)(size % ALIGN) == (short)0) return (short)0;
  return (short)(ALIGN - size % ALIGN );
}

public static short   Set_Att(short att, byte[] ref_att,
                              short off_att, short len_att, byte[] pkt, short off)
{
  short tl = (short) (len_att + 6) ;
  short tp = GetPadLength(tl)      ;

  tl= (short) (tp+tl);

  Util.setShort(pkt,off,att)  ;
  Util.setShort(pkt,(short)(off+2),tl);
  Util.setShort(pkt,(short)(off+4),tp);

  if (ref_att != null)
  Util.arrayCopy(ref_att,off_att,pkt,(short)(off+6),len_att);
  else
  Util.arrayFillNonAtomic(pkt,(short)(off+6),len_att,(byte)0);

  if (tp != (short)0)
  Util.arrayFillNonAtomic(pkt,(short)(off+6+len_att),tp,(byte)0);

  return tl ;
}
```

```
public void process(APDU apdu) throws ISOException
  { short len=(short)0, readCount=(short)0;
       short off=(short)0,pad=(short)0,len_r1=(short)0;
       short size=(short)0;

  byte[] buffer = apdu.getBuffer() ; // lecture CLA INS P1 P2 P3

  byte cla = buffer[ISO7816.OFFSET_CLA];
  byte ins = buffer[ISO7816.OFFSET_INS];
  byte P1  = buffer[ISO7816.OFFSET_P1] ;
  byte P2  = buffer[ISO7816.OFFSET_P2] ;
  byte P3  = buffer[ISO7816.OFFSET_LC] ;

  switch (ins)
              {

              case SELECT:
                size = apdu.setIncomingAndReceive();
                return;

              case INS_I1:

                     rnd.generateData(DB,off_myHIT,(short)16);

                     Util.arrayCopy(HEADER,(short)0,buffer,(short)0,(short)8);
                     Util.arrayCopy(DB,off_myHIT,buffer,(short)8,(short)16)  ;
                     Util.arrayFillNonAtomic(DB,(short)24,(short)16,(byte)0) ;
                     apdu.setOutgoingAndSend((short)0,(short)40)             ;

                     break;

              case INS_I2:

                     size = apdu.setIncomingAndReceive();

                     len = Util.makeShort((byte)0,buffer[6]);
                     len = (short)(len << 3);
                     len = (short)(len+(short)8)   ;

                     if (len != size) ISOException.throwIt(ISO7816.SW_DATA_INVALID) ;

                     size = (short)(len-(short)40);

                     // HEADER 00...08

                     // HIT-S  08...24
                     // HIT-D  24...40

       // if (Util.arrayCompare(buffer,(short)(5+24),DB,off_myHIT,(short)16) != (short) 0)
       // ISOException.throwIt(ISO7816.SW_DATA_INVALID) ;

                     Util.arrayCopy(buffer,(short)13,DB,off_rHIT,(short)16);

                     off= GetAttOffset(buffer,(short)45,size,R_T);
                     if (off==(short)-1) ISOException.throwIt(ISO7816.SW_DATA_INVALID) ;
                     len = Util.getShort(buffer,(short)(off+2));
                     pad = Util.getShort(buffer,(short)(off+4));
                     len = (short)(len-pad-6);

                     len_r1=len;Util.arrayCopy(buffer,(short)(off+6),DB,off_R1,len);

                     off= GetAttOffset(buffer,(short)45,size,HIP_T_TRANSFORM)          ;
                     if (off==(short)-1) ISOException.throwIt(ISO7816.SW_DATA_INVALID) ;
                     len = Util.getShort(buffer,(short)(off+2));
                     pad = Util.getShort(buffer,(short)(off+4));
                     len = (short)(len-pad-6);

                     // algo=Util.getShort(buffer,(short)(off+6)

                     rnd.generateData(DB,(short)(off_R1+len_r1),len_r2); // r1 || r2
```

```
            Util.arrayCopy(MyEPCCODE,(short)0,buffer,(short)0,(short)MyEPCCODE.length);
                    hmac(DB,off_R1,(short)(len_r1+len_r2),
                             buffer,(short)0,(short)MyEPCCODE.length,
                             sha1,
                             DB, off_k);

                    Util.arrayCopy(ct1,(short)0,buffer,(short)0,(short)ct1.length);
                    hmac(DB,off_k,(short)20,
                             buffer,(short)0,(short)ct1.length,
                             sha1,
                             DB, off_FT);


                     Util.arrayCopy(ct2,(short)0,buffer,(short)0,(short)ct2.length);
                     hmac(DB,off_k,(short)20,
                             buffer,(short)0,(short)ct2.length,
                             sha1,
                             DB, off_kaut);


                     Util.arrayCopy(HEADER,(short)0,buffer,(short)0,(short)HEADER.length);
                     Util.arrayCopy(DB,off_myHIT,buffer,(short)8, (short)16);
                     Util.arrayCopy(DB,off_rHIT, buffer,(short)24,(short)16);

                     off=(short)40;

                     len =
                     Set_Att(HIP_T_TRANSFORM,algo1,(short)0,(short)algo1.length,buffer,off);
                     off = (short)(off+len);

                     len = Set_Att(R_T,DB,(short)(off_R1+len_r1),len_r2,buffer,off);
                     off = (short)(off+len);

                     len = Set_Att(F_T,DB,off_FT,(short)20,buffer,off);
                     off = (short)(off+len);

                     len = Set_Att(Signature_T,null,(short)0,(short)20,buffer,off);
                     size= (short)(off+len);

                     buffer[1] = (byte) (size >>3);

                     hmac(DB,off_kaut,(short)20,
                             buffer,(short)0,size,
                             sha1,
                             buffer,(short)(off+6));

                     apdu.setOutgoingAndSend((short)0,size);


                     break;


            default:

                           ISOException.throwIt(ISO7816.SW_INS_NOT_SUPPORTED);
}  }

     /**
      * Only this class's install method should create the applet object.
      * @see APDU
      * @param apdu the incoming APDU containing the INSTALL command.
      */
     protected tag(byte[] bArray,short bOffset,byte bLength)
     {      init();
            register();                 }

     /**
      * Installs this applet.
      * @see APDU
      * @param apdu the incoming APDU containing the INSTALL command.
      * @exception ISOException with the response bytes per ISO 7816-4
      */
     public static void install( byte[] bArray, short bOffset, byte bLength )
     {
     new tag(bArray,bOffset,bLength);
```

```
        }

    public boolean select()
    {
     return true;
    }

    public void deselect()
    {
    }

    public tag()
    {  return;
    }

     // Buffer d MUST have a size greater than ld+BLOCKSIZE+DIGESTSIZE

    final static short BLOCKSIZE  =  (short)64 ;
    final static short DIGESTSIZE =  (short)20 ;
    final static short DEC =  (short)84; // 64+20

     public void  hmac
          ( byte []  k,short k_off, short lk,
            byte []  d,short d_off,short ld,
            MessageDigest md,
            byte out[], short out_off)
           {
            short i,con_len ;

            if (lk > BLOCKSIZE )
            { md.reset();
                 md.doFinal(k,k_off,lk,k,k_off);
                 lk = DIGESTSIZE ;
            }

            Util.arrayCopyNonAtomic(d,d_off,d,(short)(d_off+BLOCKSIZE),ld);

            for (i = 0 ; i < lk ; i=(short)(i+1))
            d[(short)(i+d_off)] = (byte)(k[(short)(i+k_off)] ^ (byte)0x36) ;

            Util.arrayFillNonAtomic(d,(short)(d_off+lk),(short)(BLOCKSIZE-lk),(byte)0x36);

            // KEY|DATA

            md.reset();
            md.doFinal(d,d_off,(short)(ld+BLOCKSIZE),d,(short)(d_off+ld+BLOCKSIZE));

            Util.arrayCopyNonAtomic(d,(short)(d_off+BLOCKSIZE),d,d_off,ld);

            // KEY|HASH
            for (i = 0 ; i < lk ; i=(short)(i+1))
            d[(short)(d_off+ld+i)] = (byte)(k[(short)(i+k_off)] ^ (byte)0x5C)  ;
            Util.arrayFillNonAtomic(d,(short)(d_off+ld+lk),(short)(BLOCKSIZE-lk),(byte)0x5C);

            md.reset();
            md.doFinal(d,(short)(d_off+ld),DEC,out,out_off);

           }


}
```