```
<body>

<?php
print" <body style='background: #FFCC99;'>";
print "<font face= 'courier'  size='4' >";
// define variables and set to empty values
$passwordErr = $bitsErr = $textErr = $transformErr="";
$pw = $nrOfBits = $text = $trans="";
/*
                    AES Counter Mode

        Programmed with reference to:

https://engineering.purdue.edu/kak/compsec/NewLectures/Lecture8.pdf
            http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-
38a.pdf
        I have also used some of the programming ideas from Chris
Veness' excellent
        AES section at http://www.movable-
type.co.uk/scripts/aes.html, though with
        changes and additions to enable choice of keylengths, modes
other than CTR,
        selection of nonce, incrementing the counter and to suit my
own style of programming.

       This program is controlled in functions encipher() &
decipher().
       Rijndael algorithm is performed in function getKeySchedule()
       & in function Rijndael().

       These functions have been validated using in Sections F.5
       in the NIST document mentioned above.

       No warranty of any form is offered for this program.

       Mike Cowan 27:05:2014
       */


function Rijndael($in, $word)
  {
    /*
     see
https://engineering.purdue.edu/kak/compsec/NewLectures/Lecture8.pdf
     and http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf
pages 14 & 15.

     input of 16 bytes is put into a 4x4 'state array', column by
column.

     word is the Key Schedule, made earlier, comprising 4-byte words.
There are
     44/52/60 words, dependant on key length.
    */


    $state=array(4);
    for($r=0;$r<4;$r++) $state[$r]=array(4);
    $t=0;
    for($c=0;$c<4;$c++)
      for($r=0;$r<4;$r++)
```

```
         {$state[$r][$c] = $in[$t];$t++;}


    $nrOfRounds = (count($word)/4) - 1;

    $state = addRoundKey($state, $word, 0);

    for ($round=1; $round<$nrOfRounds; $round++)
       {
        $state = subBytes($state);
        $state = shiftRows($state);
        $state = mixColumns($state);
        $state = addRoundKey($state, $word, $round);
       }

    $state = subBytes($state);
    $state = shiftRows($state);
    $state = addRoundKey($state, $word, $nrOfRounds);

  //...make 16-byte out by stripping state column by column...
    $out=array(16);
    $t=0;
    for($c=0;$c<4;$c++)
      for($r=0;$r<4;$r++)
         {$out[$t] = $state[$r][$c]; $t++;}

    return $out;
  }
 //..................................................


  function subBytes($state)
    {
    global $sBox;
    for ($r=0; $r<4; $r++)
       {
         for ($c=0; $c<4; $c++) $state[$r][$c] =
$sBox[$state[$r][$c]];
       }
    return $state;
  }

function shiftRows($state)
   {
    $copy = array(4); //temporary usage
    for ($r=1; $r<4; $r++)
      {
       for ($c=0; $c<4; $c++) $copy[$c] = $state[$r][($c+$r)%4];
       for ($c=0; $c<4; $c++) $state[$r][$c] = $copy[$c];
      }
    return $state;
  }

function mixColumns($state)
   {
      //http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf
page 17
      for ($c=0; $c<4; $c++)
        {
         $a = array(4);  // a[n] is a copy of the current input
column
```

```
         $b = array(4);  // b[n] is going to be 2 * a[n] in the
Galois Field
                         //  a[n]^b[n]   will be 3 * a[n]

            for($j=0;$j<4;$j++)
               {
                  $a[$j] = $state[$j][$c];
                  $b[$j] =  $state[$j][$c]<<1;   //left shift doubles
the value of $s[][]
                  if($b[$j]>255) {$b[$j]-=256;$b[$j]^=27;} // but if
result >255 then must
                                                                //
subtract 256 & xor with 27
               }

         $state[0][$c] = $b[0] ^ $a[1] ^ $b[1] ^ $a[2] ^ $a[3]; //
2a0 + 3a1 +  a2 +  a3;
         $state[1][$c] = $a[0] ^ $b[1] ^ $a[2] ^ $b[2] ^ $a[3]; //
a0 * 2a1 + 3a2 +  a3
         $state[2][$c] = $a[0] ^ $a[1] ^ $b[2] ^ $a[3] ^ $b[3]; //
a0 +  a1 + 2a2 + 3a3
         $state[3][$c] = $a[0] ^ $b[0] ^ $a[1] ^ $a[2] ^ $b[3]; //
3a0 +  a1 +  a2 + 2a3
        }
      return $state;
   }

function addRoundKey($state, $word, $roundNumber)
    {
      for ($r=0; $r<4; $r++)
       {
         for ($c=0; $c<4; $c++)
              {
                $nr= (4*$roundNumber) + $c; // word number
                $state[$r][$c] ^= $word[$nr][$r];
              }
       }
    return $state;
  }


 function rotate($word)
 {
    $copy = $word[0];
    for ($j=0; $j<3; $j++) $word[$j] = $word[$j+1];
    $word[3] = $copy;
    return $word;
  }



    $rCon = array(
    array(0,0,0,0),
    array(1,0,0,0),
    array(2,0,0,0),
    array(4,0,0,0),
    array(8,0,0,0),
    array(16,0,0,0),
    array(32,0,0,0),
    array(64,0,0,0),
    array(128,0,0,0),
```

```
    array(27,0,0,0),
    array(54,0,0,0) );

    $sBox = array(

99,124,119,123,242,107,111,197,48,1,103,43,254,215,171,118,202,130,20
1,125,250,89,71,240,173,212,162,175,

156,164,114,192,183,253,147,38,54,63,247,204,52,165,229,241,113,216,4
9,21,4,199,35,195,24,150,5,154,7,18,

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,

56,245,188,182,218,33,16,255,243,210,205,12,19,236,95,151,68,23,196,1
67,126,61,100,93,25,115,96,129,79,220,

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,

55,109,141,213,78,169,108,86,244,234,101,122,174,8,186,120,37,46,28,1
66,180,198,232,221,116,31,75,189,139,

138,112,62,181,102,72,3,246,14,97,53,87,185,134,193,29,158,225,248,15
2,17,105,217,142,148,155,30,135,233,

206,85,40,223,140,161,137,13,191,230,66,104,65,153,45,15,176,84,187,2
2);


//.....
......


function getKeySchedule($key)
  {
    /* see
https://engineering.purdue.edu/kak/compsec/NewLectures/Lecture8.pdf
       pages 7 and 8.
       The key is loaded into words[0->3] of 4 bytes each.
       There will be 10/12/14 rounds for 128/192/256 bit keys, which
means there
        will be 44, 52, 60 words in total.
       In each round:
       -the first word is made from the previous word processed
through function 'g'
         (see below) and then xored with the first word of the
previous round;
       -the next three words are made from the previous word xored
with the word
         in the same place from the previous round.

       The function 'g' submits copy to functions rotate, subBytes
and rCon.
      */

    global $sBox,$rCon;
```

```
    $nrOfWords = count($key)/4;       // number of 4-byte words in
key =4/6/8
    $nrOfRounds = $nrOfWords + 6;     // number of rounds =10/12/14

    $word=array(70);
    for($i=0;$i<4;$i++) $word[$i]=array(4);

    $t=0;
    for($i=0;$i<$nrOfWords;$i++)
      for($k=0;$k<4;$k++)
         {$word[$i][$k] = $key[$t];$t++;}

      //....for 128 bit key make four 4-byte words for 10 rounds.....
      // ..........numbered $word[4] to $word[43].................
      //.. up to word[51] or word[59] for 192 or 256 bit keys
resp.....
   for ($j=$nrOfWords; $j<(4*($nrOfRounds+1)); $j++)
      {
       //...make a copy of the previous four words.......
       for ($t=0; $t<4; $t++) $copy[$t] = $word[$j-1][$t];

       //...if we are dealing with the first word of a group of four
      //..... then make function g from copy.................
       if ($j % $nrOfWords == 0)
         {
           //...do rotate......
          $copy = rotate($copy);
          //.....do subBytes........
         for ($t=0; $t<4; $t++) $copy[$t] = $sBox[$copy[$t]];
          //.....do rCon..................
          for ($t=0; $t<4; $t++) $copy[$t] ^=
$rCon[$j/$nrOfWords][$t];
         }
                //...extra routine if key is 256 bits....
        else if($nrOfWords==8 && ($j-4)%8==0 )
         {
           for ($t=0; $t<4; $t++) $copy[$t] = $sBox[$copy[$t]];
         }

      //...finally  xor copy with the word from same position in last
round....
      for ($t=0; $t<4; $t++) $word[$j][$t] = $word[$j-$nrOfWords][$t]
^ $copy[$t];
     }

     return $word;
 }


//.............................................................
 function encipher($key,$ptDigits,$nrOfBits)
    {
      $keySchedule = getKeySchedule($key);

    // initialise 1st 8 bytes of counter with nonce for repeated
later use
    $counter = array();
    $sede = floor(microtime(true)*1000);
    srand($sede);
    for($j=0;$j<8;$j++)   $counter[$j]=rand(0,256);
```

```
    // and convert it to a string to go on the front of the
ciphertext
    $nonce = '';
    for ($j=0; $j<8; $j++) $nonce .= chr($counter[$j]);

                //fixed nonce
                /* $nonce="";
                 $counter=[61,62,63,64,65,66,67,68];
//<<<<<<<<<<<<<<<<<<<<<<<
                 $nonce = '';for($i=0; $i<8; $i++) $nonce .=
chr($counter[$i]);//<<<<<<<<<<<<<<<<<<<   */

    $lenPlaintext=count($ptDigits);
    $ciphertext=""; $nr=0;

    $ctr=0;
    for($j=0;$j<$lenPlaintext;$j+=16)
       {
          //..first 8 bytes of counter are the nonce; now append 8
bytes of ctr
          $x=$ctr;
          for($m=0;$m<8;$m++)
            {
              $y=$x%256; $counter[15-$m]=$y;
              $x=$x/256;
            }


         //...do Rijndael encryption with counter and keySchedule....
         //       to produce 16 bytes of encryption
           $encryption =  Rijndael($counter, $keySchedule);
        //...xor plaintext with encryption
          $end = $j+16;
          if($end>$lenPlaintext) $nr=$lenPlaintext-$j;
          else $nr=16;

          for ($k=0; $k<$nr; $k++)
           {
             // ..xor encrypted digit with pt digit to get cipher
digit ...
             $xor = $encryption[$k] ^ $ptDigits[$j+$k];
             $ciphertext=$ciphertext.chr($xor); // now have ct
character

           }
         //...increment counter....
           $pos=15;
           $tot= $counter[$pos]+1;
           $counter[$pos]=$tot%256;
           $carry = floor($tot/256);
           while($carry>0)
                {
                 $pos--;
                 $tot=$counter[$pos]+1;
                 $counter[$pos]=$tot%256;
                 $carry = floor($tot/256);
                }
       }

      //...prepend nonce to ciphertext.....
      $mk=$nonce.$ciphertext;
```

```
      $ciphertext=$mk;

     $ct = base64_encode($ciphertext);
     return $ct;
  }

//.............................................................

 function decipher($ciphertext, $key, $nrOfBits)
  {

      $keySchedule = getKeySchedule($key);

    // recover nonce from 1st element of ciphertext
      $counter = array();
      for ($j=0; $j<8; $j++) $counter[$j] =
ord(substr($ciphertext,$j,1));

      $plainex=array();
      $ctr=0; $t=0; $lenCiphertext=strlen($ciphertext);
      for($j=8;$j<$lenCiphertext;$j+=16)
       {
        //..make counter of 8-byte nonce followed by 8-byte block
number
         $x=$ctr;
         for($m=0;$m<8;$m++)
            {
              $y=$x%256; $counter[15-$m]=$y;
              $x=$x/256;
            }

         //...do Rijndael encryption with counter and keySchedule....
         //       to produce 16 bytes of encryption
         $encryption =  Rijndael($counter, $keySchedule);

          $end = $j+16;
          if($end>$lenCiphertext) $nr=$lenCiphertext-$j;
          else $nr=16;

          for ($k=0; $k<$nr; $k++)
            {
                $plainex[$t]= $encryption[$k] ^
ord(substr($ciphertext,($j+$k),1));
                $t++;
             }
         //...increment counter....
           $pos=15;
           $tot= $counter[$pos]+1;
           $counter[$pos]=$tot%256;
           $carry = floor($tot/256);
           while($carry>0)
                {
                 $pos--;
                 $tot=$counter[$pos]+1;
                 $counter[$pos]=$tot%256;
                 $carry = floor($tot/256);
                }
     }

      return($plainex);
  }
```

```
//-------------------------------------------------------------
//.....MAIN........


   if ($_SERVER["REQUEST_METHOD"] == "POST")
  {
     if (empty($_POST["password"]))
         {
          $passwordErr = "Password is required";
         }
     else {
           $pw = test_input($_POST["password"]);
          }


     if (empty($_POST["bits"]))
         {
          $bitsErr = "bits is required";
         }
     else {
          $nrOfBits = test_input($_POST["bits"]);
         }


     if (empty($_POST["text"]))
        {
         $textErr = "text is required";
        }
    else
       {
        $txt = test_input($_POST["text"]);
       }

    if (empty($_POST["transform"]))
         {
          $transformErr = "transform is required";
         }
     else {
          $trans = test_input($_POST["transform"]);
         }

  }

function test_input($data)
   {
     $data = trim($data);
     $data = stripslashes($data);
     $data = htmlspecialchars($data,ENT_IGNORE);
     return $data;
   }

 function digitsFromText($txt)
   {
     $digits=array(); $t=0;
     for($j=0;$j<strlen($txt);$j++)
       if(substr($txt,$j,1)=='&')
        {
         if(substr($txt,$j,5)==="&amp;")
{$digits[$t]=38;$t++;$jump=4; }
```

```
         if(substr($txt,$j,4)=="&lt;")   {$digits[$t]=60; $t++;
$digits[$t]=32;$t++;$jump=3; }
         if(substr($txt,$j,4)=="&gt;")
{$digits[$t]=62;$t++;$jump=3; }
         if(substr($txt,$j,6)=="&quot;")
{$digits[$t]=34;$t++;$jump=5;}
         if(substr($txt,$j,6)=="&#039;")  {$tt[$t]=39;$t++;$jump=5;}
         $j=$j+$jump;
        }
      else {$digits[$t]= ord(substr($txt,$j,1));$t++;}
      return $digits;
   }

 function extendKey($key,$nrOfBits)
    {

       //...ensure password digits to required number of bytes..
      $len_password=count($key);
      $nrOfBytes= $nrOfBits/8;
      $diff=$nrOfBytes-$len_password;
      for($j=0;$j<$diff;$j++)
            {
             $x=$j%$len_password;
             $key[$len_password+$j]=$key[$j];
            }
       return $key;
    }

?>

<h2>AES CTR MODE</h2>
<p><span class="error">* required field.</span></p>
<form method="post" action="<?php echo
htmlspecialchars($_SERVER["PHP_SELF"]);?>">

   password:
   <input type="text" name="password" value="<?php echo $pw;?>">
   <span class="error">* <?php echo $passwordErr;?></span>
   <br><br>

    Number of Bits:
   <input type="radio" name="bits" <?php if (isset($nrOfBits) &&
$nrOfBits=="128") echo "checked";?>  value="128">128
   <input type="radio" name="bits" <?php if (isset($nrOfBits) &&
$nrOfBits=="192") echo "checked";?>  value="192">192
   <input type="radio" name="bits" <?php if (isset($nrOfBits) &&
$nrOfBits=="256") echo "checked";?>  value="256">256
   <span class="error">* <?php echo $bitsErr;?></span>
   <br><br>

   input: <br><textarea name="text" rows="10" cols="80"><?php echo
$txt;?></textarea>
   <span class="error">* <?php echo $textErr;?></span>
   <br><br>

   transform:
   <input type="radio" name="transform" <?php if (isset($trans) &&
$trans=="encipher") echo "checked";?>  value="encipher">encipher
   <input type="radio" name="transform" <?php if (isset($trans) &&
$trans=="decipher") echo "checked";?>  value="decipher">decipher
   <span class="error">* <?php echo $transformErr;?></span>
```

```
   <br><br>


   <input type="submit" name="submit" value="Submit">
</form>

<?php

   if(strlen($pw)==0 || strlen($txt)==0) $bad=1;
   else $bad=0;

       if($trans=='encipher' && $bad==0)
          {
           $key = array();
           $key=digitsFromText($pw);
           $key =extendKey($key,$nrOfBits);
           $ptDigits=digitsFromText($txt);
           $ct=encipher($key,$ptDigits,$nrOfBits);
           $out=$ct;
         }

  if($trans=='decipher' && $bad==0)
      {
        //...check that ciphertext has only permitted characters...
        $good = "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789+/=";
        for($j=0;$j<strlen($txt);$j++)
           {
            $ch=substr($txt,$j,1);
            $ch=strtoupper($ch);
            $bad=1;
            for($k=0;$k<39;$k++)
                if($ch==substr($good,$k,1)) $bad=0;
             if(ord($ch) ==10 || ord($ch)==13) $bad=0;
            if($bad==1)
               {
                 print "<font color = 'red' font face= 'arial'
size='5' >";
                 print "<br> Bad character found in Ciphertext, not
present in base64 encoding <br>";
                 print" which uses only letters of alphabet, numbers
0 to 9 and + or / <br>";
                 print"ciphertext may end in = or == <br>";
                 print " I am aborting this run <br>";
                 break;
              }
           }

        $ciphertext = base64_decode($txt);


        if($bad==0)
            {

              $key = array();
              $key=digitsFromText($pw);
              $key =extendKey($key,$nrOfBits);
              $plainex = decipher($ciphertext, $key, $nrOfBits);
              $out="";
              for($j=0;$j<count($plainex);$j++)
                 if($plainex[$j]==10) $out.="\n";
                 else $out.=chr($plainex[$j]);
```

```
            }
    }


print "<font color = 'black' font face= 'arial'  size='4' Error in
Ciphertext <br>";

//note: version mk1 of 13th may 2014

?>

output:
 <br><textarea name="text" rows="10" cols="80"><?php echo
$out;?></textarea>
   <br><br>
 <h5>
[validated with NIST input & output data, see:<br>
http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf]<br><br>
M J Cowan 2014    </h5>

</body>
</html>
```