OpenVMS Technical
Journal V15

Implementing Triple DES (TCBC) on OpenVMS

Rupesh Shantamurty

Implementing Triple DES (TCBC) on OpenVMS 1
Introduction 2
tdea_vms.c 3
Summary 7

Introduction

Data Encryption Standard (DES) has been supported by OpenVMS since Version 7.3 and support for Advanced Encryption Standard (AES) was added in OpenVMS Version 8.3. Both these encryption modes are now available along with the operating system. Triple Data Encryption Standard, also known as 3DES or TDEA was introduced as a temporary replacement for DES till AES was made available.
Using DES three times consecutively is 3DES. Following are the four modes of DES operation:

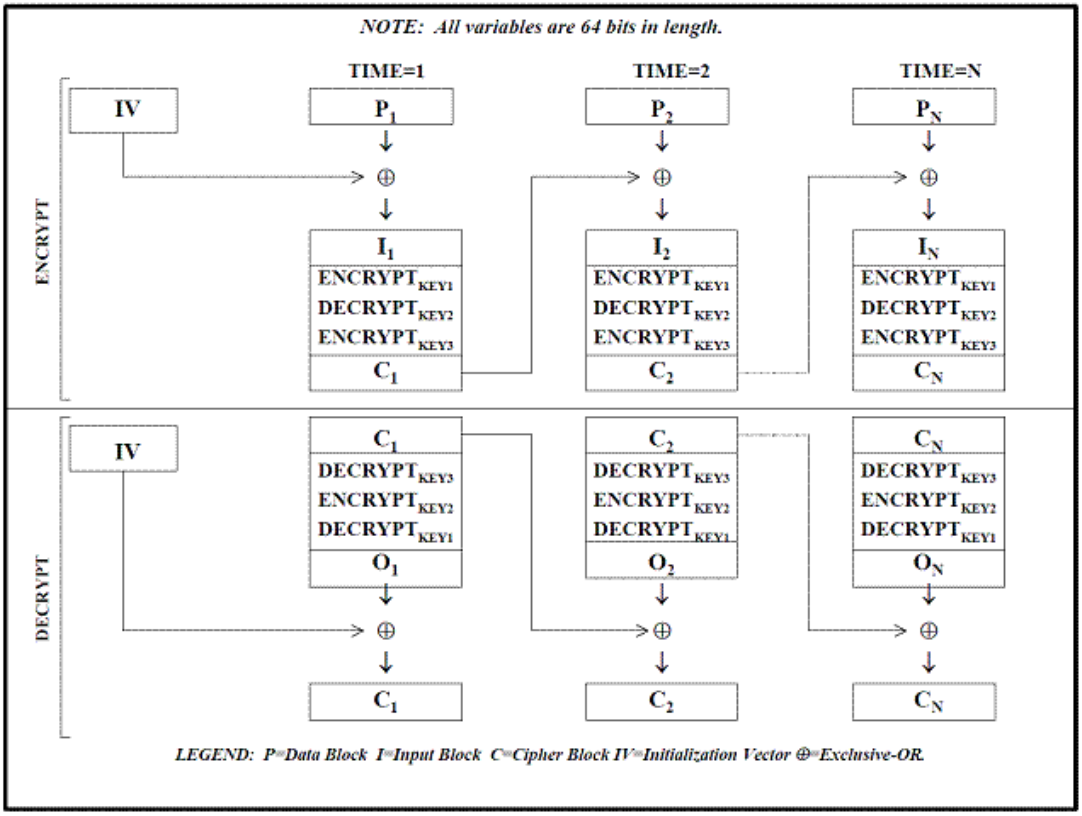
- Electronic Codebook (ECB) mode
- Cipher Block Chaining (CBC) mode
- Cipher Feedback (CFB) mode
- Output Feedback (OFB) mode

For more information about DES, see the NIST document[1]. 3DES has seven modes of operation, four of which have been derived from DES.

OpenVMS does not provide direct Application Programming Interface (API) to encrypt or decrypt using 3DES. But some OpenVMS users have used the available DES APIs to encrypt and decrypt. There are a few subtle points that should be considered when implementing 3DES using the DES APIs on OpenVMS.

This article demonstrates how to encrypt and decrypt programmatically in C, using the example of Cipher Block Chaining (CBC) mode of operation. Coding has been done based on the schema presented in the NIST document explaining TDEA[2] as follows:

TDEA Cipher Block Chaining (TCBC) Mode



[1] DES http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf

[2] TDEA http://csrc.nist.gov/publications/nistpubs/800-20/800-20.pdf

The most important thing to remember in this implementation is to take just 64 bits of data at a time. For the encryption phase the cipher output of the first set of 64bits is used as the IV (Initialization Vector) for the second set of 64 bits and so on. For decryption the IV will have to be used only in the last decrypt operation using key1 and the IV for the next set of 64 bits is the first set of 64bits of the cipher text.

Below is the program in C programming language for encryption using TDEA CBC mode

tdea_vms.c

```
---------------
/* The cipher text generated can be verified using the Cryptosys API
 * ( www.cryptosys.net ,the Personal Edition is free for personal
 *   use on a stand-alone computer)
 */
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <descrip>
#include <ssdef>
#include <time>
#include <libdtdef>
#include <lib$routines.h>
#include "ENCRYPT$EXAMPLES:encrypt_def.h"
#include <string.h>

#define KEY_VALUE_FLAG 1
char cbc_out[8];
char *des_cbc_once(char *data,char *key, char *iv,int option);
void main ()
{
  unsigned char input_data[32], my_data[32], decryp_data[32];

  /* Note: The key has been hardcoded to avoid any endian conflicts,
   * Refer to OpenVMS FAQ for more information on endian considerations.
   */

  unsigned char mykey[24]= { 0x01,0x23,0x45,0x67,0x89,0xab,0xcd,0xef,
                             0xfe,0xdc,0xba,0x98,0x76,0x54,0x32,0x10,
                             0xab,0xcd,0xef,0x01,0x23,0x45,0x67,0x89
                            };

  /* The initialization vector has been hardcoded */

  unsigned char iv[8]={0x12,0x34,0x56,0x78,
                       0x90,0xab,0xcd,0xef};

  static unsigned char cbc_iv[8];

  unsigned char * output;
  unsigned char cipher[32],des_data[8],key1[8],key2[8],key3[8];

  int i,input_length,noof64bit;

  memset(my_data,0,32);
  printf("\nType the text that needs to be encrypted(max 32 chars) : ");
  gets((char *)input_data);
  input_length = strlen((char *)input_data);

  noof64bit = ceil((float)input_length/8);
```

```
memcpy(my_data,input_data,noof64bit*8);

printf("\n The Input text in hex is : \n");
for (i=0;i<noof64bit*8;i++)
   printf("%.2X",(unsigned char *)my_data[i]);

/* Set the IV to it's initial value */
memcpy(cbc_iv,iv,8);

printf("\n The IV in hex is : ");
for (i=0;i<8;i++)
   printf("%.2X",(unsigned char *)cbc_iv[i]);

printf("\n The 192 bit Key in hex is : \n");
for (i=0;i<24;i++)
   printf("%.2X",(unsigned char *)mykey[i]);

memcpy(key1,mykey,8);
memcpy(key2,mykey+8,8);
memcpy(key3,mykey+16,8);

/* Encryption Phase */

for (i=1;i<=noof64bit;i++)
{
   memset(des_data,0,8);
   memcpy( des_data,my_data+((i-1)*8),8);

   /* The Core part of encryption in TCBC mode */
   /* First encrypt using key1 , use IV also */
   output = (unsigned char *)des_cbc_once((char *)des_data,
                                          (char *)key1,
                                          (char *)cbc_iv,1);

   /* Decrypt the output of previous step using key2 */
   output = (unsigned char *)des_cbc_once((char *)output,
                                          (char *)key2,NULL,0);

   /* Finally encrypt the output of previous step using key3 */
   output = (unsigned char *)des_cbc_once((char *)output,
                                          (char *)key3,NULL,1);

   memcpy(cipher+((i-1)*8),output,8);

   memcpy(cbc_iv,output,8);
}
printf("\n The Cipher text in hex is : \n");
for (i=0;i<noof64bit*8;i++)
   printf("%.2X",(unsigned char *)cipher[i]);

/* Decryption Phase */

/* Set the IV to it's initial value */
memcpy(cbc_iv,iv,8);

for (i=1;i<=noof64bit;i++)
{
   memset(des_data,0,8);

   memcpy( des_data, cipher+((i-1)*8),8);
```

```
      /* The Core part of deryption in TCBC mode */

      /* First decrypt using key3 , with no IV */
      output = (unsigned char *)des_cbc_once((char *)des_data,
                                             (char *)key3,NULL,0);

      /* Encrypt the output of previous step using key2 */
      output = (unsigned char *)des_cbc_once((char *)output,
                                             (char *)key2,NULL,1);

      /* Finally decrypt the output of previous step using key1,
         use IV also */
      output = (unsigned char *)des_cbc_once((char *)output,
                                             (char *)key1,
                                             (char *)cbc_iv,0);
      memcpy(decryp_data+((i-1)*8),output,8);

      memcpy(cbc_iv, des_data,8);
   }
   printf("\n The Deciphered text in hex is : \n");
   for (i=0;i<noof64bit*8;i++)
      printf("%.2X",(unsigned char *)decryp_data[i]);

   printf("\n The Deciphered text is : \n");
   puts((char*) decryp_data);

} /* End of main */

/* Function to do CBC, using DES API, once */
char * des_cbc_once( char *data,char *key, char *iv,int option)
{
    char int_data[8],int_key[8],int_iv[8];
    unsigned long  context     = 0;
    unsigned long  key_type    = KEY_VALUE_FLAG ;
    int            encr_out_len = 0;
    long           status      = 0;

    memcpy(int_data,data,8);
    memcpy(int_key,key,8);
    if(iv!=NULL)
    memcpy(int_iv,iv,8);

    /* Define input, output, algo and key descriptors */
    $DESCRIPTOR(input_desc,     "");
    $DESCRIPTOR(output_desc,    "");
    $DESCRIPTOR(algo_desc,"DESCBC");
    $DESCRIPTOR(key_buf_desc, "");

    input_desc.dsc$b_dtype = DSC$K_DTYPE_NU ;
                                    /* Numeric String Unsigned    */
    input_desc.dsc$a_pointer = (char *)&int_data;
    input_desc.dsc$w_length  = 8;

    output_desc.dsc$b_dtype = DSC$K_DTYPE_NU ;
    output_desc.dsc$a_pointer = (char *)&cbc_out;
    output_desc.dsc$w_length = 8;

    key_buf_desc.dsc$b_dtype = DSC$K_DTYPE_NU ;
    key_buf_desc.dsc$a_pointer = (char *)&int_key;
    key_buf_desc.dsc$w_length  = 8 ;
```

```
    status  = encrypt$init ( &context , &algo_desc , &key_type,
                             &key_buf_desc , int_iv );

    if (!(status & 1))
    {
        printf ("\nencrypt$init() function failed\n");
        lib$signal (status);
        return (0);
    }
    memset(cbc_out,0,8);

    /* If option is 1 then do encryption, else do decryption */

    if(option==1)
    {
       status = encrypt$encrypt( &context, &input_desc,
                                 &output_desc, &encr_out_len );
       if (!(status & 1))
       {
         printf ("\n encrypt$encrypt() function failed.\n");
         lib$signal (status);
         return (0);
       }
    }
    else
    {
       status = encrypt$decrypt( &context, &input_desc ,
                                 &output_desc, &encr_out_len );
       if (!(status & 1))
       {
         printf ("\n encrypt$decrypt() function failed.\n");
         lib$signal (status);
         return (0);
       }
    }

    status = encrypt$fini (&context);
    if (!(status & 1))
    {
        printf ("\n encrypt$fini() function failed.\n");
        lib$signal (status);
        return (0);
    }
    return cbc_out;
}
```

Sample output

```
$ run TDEA_VMS
Type the text that needs to be encrypted(max 32 chars) : Now is the time
for all
 The Input text in hex is :
4E6F77206973207468652074696D6520666F7220616C6C20
 The IV in hex is : 1234567890ABCDEF
 The 192 bit Key in hex is :
0123456789ABCDEFFEDCBA9876543210ABCDEF0123456789
 The Cipher text in hex is :
80B31486E9FE855A033837A54A4DDF1A97E7D083F1FD8269
 The Deciphered text in hex is :
4E6F77206973207468652074696D6520666F7220616C6C20
 The Deciphered text is :
Now is the time for all
$
```

Summary

The example program has demonstrated how easy it is to implement TCBC mode of 3DES on OpenVMS. You can also implement other modes of 3DES using a similar method. For more information, please refer to the NIST documentation.