*************Draft*************

crypto notes 05/21/20
Starting with TFTP_Template
Testing Electronic Codebook (ECB) & AES Cipher Block Chaining (CBC)

*************Draft*************

Latest work

Hello All,

Set the Path
. ./fpc.sh

Compile:
fpc -vi -B -Tultibo -Parm -CpARMV7A -WpRPI2B @/home/devel/ultibo/core/fpc/bin/RPI2.CFG -O2 test_crypto.lpr

Transfer kernel7.img to Ultibo System
./upker7.sh sleep 10

tftp 192.168.1.245
tftp> binary
tftp> get test0513.txt
Received 1541 bytes in 0.0 seconds

This is what gets written to the file test0513.txt

GCM
Strplaintext
Four score and s
even years ago o
ur fathers broug
ht forth on this
StrIV
000102030405060708090a0b0c0d0e0f
StrAAD
000102030405060708090a0b0c0d0e0f
Key
4e6f772077652061726520656e676167
798c536d4a8351dc7763d79d6434dd79
c9f83b8c04349cd0e3284367f629ac13
Enc
e5b87ad44492f895d009a769b3a3c338
372276fdf099292a996b6e5a3c06e1be
Dec
466f75722073636f726520616e642073
6576656e207965617273206167 6f206f

I tested 2 blocks of 128Bit using Galois/Counter Mode (GCM) with the following code from

https://github.com/develone/Ultibo_Projects/blob/master/test_crypto/RPi2/test_crypto.lpr

Key for first block

Ascii	Hex
{Now we are engag}	4e6f77207765206172652065 6e676167

Key for 2nd block which was the AESGCMTag from the first block

hex

798c536d4a8351dc7763d79d6434dd79

Both the AESGCMIV & AESGCMAAD were the same

Hex

00010203040506070809 0a0b0c0d0e0f

Ascii	Hex
Four score and s	466f75722073636f726520616e642073
even years ago o	6576656e207965617273206167 6f206f

```
GCM1.Strplaintext[0]:='Four score and s';

  GCM1.Strplaintext[1]:='even years ago o';

  GCM1.Strplaintext[2]:='ur fathers broug';

  GCM1.Strplaintext[3]:='ht forth on this';

  AESGCMKey:=AllocMem(AES_KEY_SIZE128);
  {Now we are engag}

StringToBytes('4e6f77207765206172652065 6e676167',PByte(AESGCMKey),AES_KEY_SIZE12
8);
  GCM1.StrKeyHex[0]:=BytesToString(AESGCMKey,AES_BLOCK_SIZE);

  AESGCMTag:=AllocMem(AES_BLOCK_SIZE);
  AESGCMIV:=AllocMem(AES_BLOCK_SIZE);
  AESGCMAAD:=AllocMem(AES_BLOCK_SIZE);

StringToBytes('00010203040506070809 0a0b0c0d0e0f',PByte(AESGCMIV),AES_BLOCK_SIZE);
  GCM1.StrIV:=BytesToString(AESGCMIV,AES_BLOCK_SIZE);

StringToBytes('00010203040506070809 0a0b0c0d0e0f',PByte(AESGCMAAD),AES_BLOCK_SIZ
E);
  GCM1.StrAAD:=BytesToString(AESGCMAAD,AES_BLOCK_SIZE);
  AESGCMData:=AllocMem(AES_BLOCK_SIZE);


StringToBytes('466f75722073636f726520616e642073',PByte(AESGCMData),AES_BLOCK_SIZE
);

  ConsoleWindowWriteLn (RightWindow, 'Inputs1');
```

```
  ConsoleWindowWriteLn (RightWindow, 'Key: ' +  GCM1.StrKeyHex[0]);
  ConsoleWindowWriteLn (RightWindow, 'IV: ' +  GCM1.StrIV);
  ConsoleWindowWriteLn (RightWindow, 'AAD: ' +  GCM1.StrAAD);
  ConsoleWindowWriteLn (RightWindow, 'Data: ' +
BytesToString(AESGCMData,AES_BLOCK_SIZE));
  ConsoleWindowWriteLn (RightWindow, 'Tag: ' +
BytesToString(AESGCMTag,AES_BLOCK_SIZE));
  if
AESGCMEncryptData(AESGCMKey,AES_KEY_SIZE128,AESGCMIV,AESGCMAAD,AESGC
MData,AESGCMData,AES_BLOCK_SIZE,AES_BLOCK_SIZE,AES_BLOCK_SIZE,AESGCMT
ag) then
    begin
      GCM1.StrEnc[0]:=BytesToString(AESGCMData,AES_BLOCK_SIZE);
      ConsoleWindowWriteLn (RightWindow, 'GCMEncrypt ok ' );
    end
  else
    begin
    ConsoleWindowWriteLn (RightWindow, 'GCMEncrypt failed');
    end;

  ConsoleWindowWriteLn (RightWindow, 'Key: ' +  GCM1.StrKeyHex[0]);
  ConsoleWindowWriteLn (RightWindow, 'IV: ' +
BytesToString(AESGCMIV,AES_BLOCK_SIZE));
  ConsoleWindowWriteLn (RightWindow, 'ADD: ' +
BytesToString(AESGCMAAD,AES_BLOCK_SIZE));
  ConsoleWindowWriteLn (RightWindow, 'Data: ' +  GCM1.StrEnc[0]);
  GCM1.StrKeyHex[1]:=BytesToString(AESGCMTag,AES_BLOCK_SIZE);
  {AESGCMTag becomes the key for the next block}
  ConsoleWindowWriteLn (RightWindow, 'Tag: ' + GCM1.StrKeyHex[1] );
   FreeMem(AESGCMIV);
   FreeMem(AESGCMAAD);
   FreeMem(AESGCMData);
   FreeMem(AESGCMTag);

  AESGCMKey:=AllocMem(AES_KEY_SIZE128);
  StringToBytes(GCM1.StrKeyHex[0],PByte(AESGCMKey),AES_KEY_SIZE128);
  AESGCMTag:=AllocMem(AES_BLOCK_SIZE);

  AESGCMIV:=AllocMem(AES_BLOCK_SIZE);
  AESGCMAAD:=AllocMem(AES_BLOCK_SIZE);
  StringToBytes(GCM1.StrIV,PByte(AESGCMIV),AES_BLOCK_SIZE);

  StringToBytes(GCM1.StrAAD,PByte(AESGCMAAD),AES_BLOCK_SIZE);

  AESGCMData:=AllocMem(AES_BLOCK_SIZE);
  StringToBytes(GCM1.StrEnc[0],PByte(AESGCMData),AES_BLOCK_SIZE);

  ConsoleWindowWriteLn (RightWindow, 'Inputs Decrypt');
  ConsoleWindowWriteLn (RightWindow, 'Key: ' +  GCM1.StrKeyHex[0]);
  ConsoleWindowWriteLn (RightWindow, 'IV: ' +  GCM1.StrIV);
  ConsoleWindowWriteLn (RightWindow, 'AAD: ' +  GCM1.StrAAD);
  ConsoleWindowWriteLn (RightWindow, 'Data: ' +  GCM1.StrEnc[0]);
```

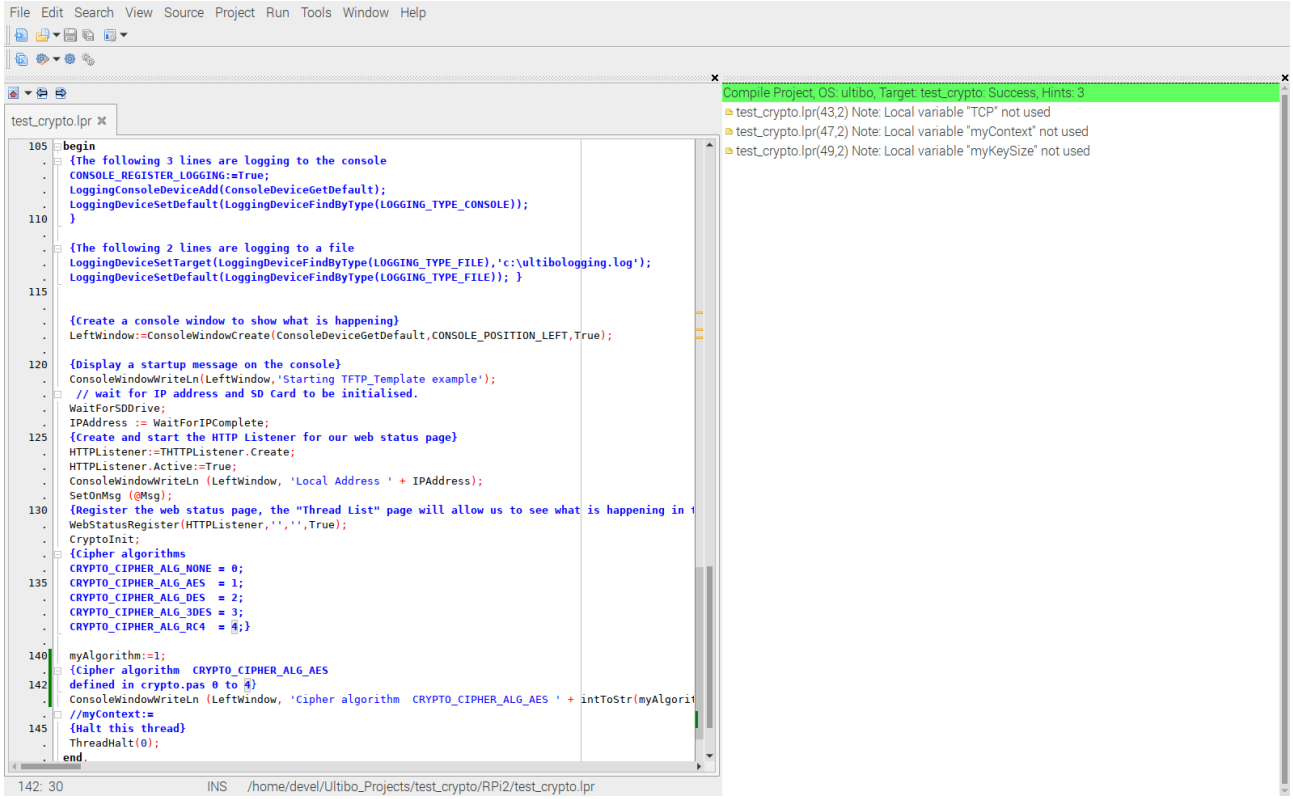
```
  ConsoleWindowWriteLn (RightWindow, 'Tag: ' +
BytesToString(AESGCMTag,AES_BLOCK_SIZE));


  ConsoleWindowWriteLn (RightWindow,'data to decrypt: '+GCM1.StrEnc[0]);
  if
AESGCMDecryptData(AESGCMKey,AES_KEY_SIZE128,AESGCMIV,AESGCMAAD,AESGC
MData,AESGCMData,AES_BLOCK_SIZE,AES_BLOCK_SIZE,AES_BLOCK_SIZE,AESGCMT
ag) then
    begin
      GCM1.StrDec[0]:=BytesToString(AESGCMData,AES_BLOCK_SIZE);

      ConsoleWindowWriteLn (RightWindow, 'GCMDecrypt ok');
    end
  else
    begin
    ConsoleWindowWriteLn (RightWindow, 'GCMDecrypt failed');
    end;
  {Since the AESGCMDecryptData is failing but the AESGCMTag is returning the  Decrypted}
  GCM1.StrDec[0]:=BytesToString(AESGCMData,AES_BLOCK_SIZE);
  ConsoleWindowWriteLn (RightWindow, 'Outputs Decrypt');
  ConsoleWindowWriteLn (RightWindow, 'Key: ' +  GCM1.StrKeyHex[0]);
  ConsoleWindowWriteLn (RightWindow, 'IV: ' +  GCM1.StrIV);
  ConsoleWindowWriteLn (RightWindow, 'ADD: ' +  GCM1.StrAAD);
  ConsoleWindowWriteLn (RightWindow, 'Data: ' +  GCM1.StrDec[0]);
  ConsoleWindowWriteLn (RightWindow, 'Data: ' +
BytesToString(AESGCMData,AES_BLOCK_SIZE));
  ConsoleWindowWriteLn (RightWindow, 'Tag: ' +
BytesToString(AESGCMTag,AES_BLOCK_SIZE));

   FreeMem(AESGCMIV);
   FreeMem(AESGCMAAD);
   FreeMem(AESGCMData);
   FreeMem(AESGCMTag);
{************end of 1st ***************}
{************start of 2nd ***************}
  AESGCMKey:=AllocMem(AES_KEY_SIZE128);
  {798c536d4a8351dc7763d79d6434dd79}
  StringToBytes(GCM1.StrKeyHex[1],PByte(AESGCMKey),AES_KEY_SIZE128);
  //GCM1.StrKeyHex[1]:=BytesToString(AESGCMKey,AES_BLOCK_SIZE);

  AESGCMTag:=AllocMem(AES_BLOCK_SIZE);
  AESGCMIV:=AllocMem(AES_BLOCK_SIZE);
  AESGCMAAD:=AllocMem(AES_BLOCK_SIZE);

StringToBytes('000102030405060708090a0b0c0d0e0f',PByte(AESGCMIV),AES_BLOCK_SIZE);
  GCM1.StrIV:=BytesToString(AESGCMIV,AES_BLOCK_SIZE);

StringToBytes('000102030405060708090a0b0c0d0e0f',PByte(AESGCMAAD),AES_BLOCK_SIZ
E);
  GCM1.StrAAD:=BytesToString(AESGCMAAD,AES_BLOCK_SIZE);
```

```
  AESGCMData:=AllocMem(AES_BLOCK_SIZE);


StringToBytes('6576656e207965617273206167 6f206f',PByte(AESGCMData),AES_BLOCK_SIZE
);

  ConsoleWindowWriteLn (RightWindow, 'Inputs1');
  ConsoleWindowWriteLn (RightWindow, 'Key: ' +  GCM1.StrKeyHex[1]);
  ConsoleWindowWriteLn (RightWindow, 'IV: ' +  GCM1.StrIV);
  ConsoleWindowWriteLn (RightWindow, 'AAD: ' +  GCM1.StrAAD);
  ConsoleWindowWriteLn (RightWindow, 'Data: ' +
BytesToString(AESGCMData,AES_BLOCK_SIZE));
  ConsoleWindowWriteLn (RightWindow, 'Tag: ' +
BytesToString(AESGCMTag,AES_BLOCK_SIZE));
  if
AESGCMEncryptData(AESGCMKey,AES_KEY_SIZE128,AESGCMIV,AESGCMAAD,AESGC
MData,AESGCMData,AES_BLOCK_SIZE,AES_BLOCK_SIZE,AES_BLOCK_SIZE,AESGCMT
ag) then
    begin
      GCM1.StrEnc[1]:=BytesToString(AESGCMData,AES_BLOCK_SIZE);
      ConsoleWindowWriteLn (RightWindow, 'GCMEncrypt ok ' );
    end
  else
    begin
    ConsoleWindowWriteLn (RightWindow, 'GCMEncrypt failed');
    end;

  ConsoleWindowWriteLn (RightWindow, 'Key: ' +  GCM1.StrKeyHex[1]);
  ConsoleWindowWriteLn (RightWindow, 'IV: ' +
BytesToString(AESGCMIV,AES_BLOCK_SIZE));
  ConsoleWindowWriteLn (RightWindow, 'ADD: ' +
BytesToString(AESGCMAAD,AES_BLOCK_SIZE));
  ConsoleWindowWriteLn (RightWindow, 'Data: ' +  GCM1.StrEnc[1]);
  GCM1.StrKeyHex[2]:=BytesToString(AESGCMTag,AES_BLOCK_SIZE);
  {AESGCMTag becomes the key for the next block}
  ConsoleWindowWriteLn (RightWindow, 'Tag: ' + GCM1.StrKeyHex[1] );
   FreeMem(AESGCMIV);
   FreeMem(AESGCMAAD);
   FreeMem(AESGCMData);
   FreeMem(AESGCMTag);

  AESGCMKey:=AllocMem(AES_KEY_SIZE128);
  StringToBytes(GCM1.StrKeyHex[1],PByte(AESGCMKey),AES_KEY_SIZE128);
  AESGCMTag:=AllocMem(AES_BLOCK_SIZE);

  AESGCMIV:=AllocMem(AES_BLOCK_SIZE);
  AESGCMAAD:=AllocMem(AES_BLOCK_SIZE);
  StringToBytes(GCM1.StrIV,PByte(AESGCMIV),AES_BLOCK_SIZE);

  StringToBytes(GCM1.StrAAD,PByte(AESGCMAAD),AES_BLOCK_SIZE);

  AESGCMData:=AllocMem(AES_BLOCK_SIZE);
```

```
  StringToBytes(GCM1.StrEnc[1],PByte(AESGCMData),AES_BLOCK_SIZE);

  ConsoleWindowWriteLn (RightWindow, 'Inputs Decrypt');
  ConsoleWindowWriteLn (RightWindow, 'Key: ' +  GCM1.StrKeyHex[1]);
  ConsoleWindowWriteLn (RightWindow, 'IV: ' +  GCM1.StrIV);
  ConsoleWindowWriteLn (RightWindow, 'AAD: ' +  GCM1.StrAAD);
  ConsoleWindowWriteLn (RightWindow, 'Data: ' +  GCM1.StrEnc[1]);

  ConsoleWindowWriteLn (RightWindow, 'Tag: ' +
BytesToString(AESGCMTag,AES_BLOCK_SIZE));


  ConsoleWindowWriteLn (RightWindow,'data to decrypt: '+GCM1.StrEnc[1]);
  if
AESGCMDecryptData(AESGCMKey,AES_KEY_SIZE128,AESGCMIV,AESGCMAAD,AESGC
MData,AESGCMData,AES_BLOCK_SIZE,AES_BLOCK_SIZE,AES_BLOCK_SIZE,AESGCMT
ag) then
    begin
      GCM1.StrDec[1]:=BytesToString(AESGCMData,AES_BLOCK_SIZE);

      ConsoleWindowWriteLn (RightWindow, 'GCMDecrypt ok');
    end
  else
    begin
    ConsoleWindowWriteLn (RightWindow, 'GCMDecrypt failed');
    end;
  {Since the AESGCMDecryptData is failing but the AESGCMTag is returning the  Decrypted}
  GCM1.StrDec[1]:=BytesToString(AESGCMData,AES_BLOCK_SIZE);
  ConsoleWindowWriteLn (RightWindow, 'Outputs Decrypt');
  ConsoleWindowWriteLn (RightWindow, 'Key: ' +  GCM1.StrKeyHex[1]);
  ConsoleWindowWriteLn (RightWindow, 'IV: ' +  GCM1.StrIV);
  ConsoleWindowWriteLn (RightWindow, 'ADD: ' +  GCM1.StrAAD);
  ConsoleWindowWriteLn (RightWindow, 'Data: ' +  GCM1.StrDec[1]);
  ConsoleWindowWriteLn (RightWindow, 'Data: ' +
BytesToString(AESGCMData,AES_BLOCK_SIZE));
  ConsoleWindowWriteLn (RightWindow, 'Tag: ' +
BytesToString(AESGCMTag,AES_BLOCK_SIZE));

   FreeMem(AESGCMIV);
   FreeMem(AESGCMAAD);
   FreeMem(AESGCMData);
   FreeMem(AESGCMTag);

 {************end of 2nd ***************}
```

Started with the file from **“TFTP_Template.lpr”** to creare **“test_crypto.lpr”** & **“test_crypto.lpi”**
In addition this needs **uTFTP.pas, upker7.sh, and cmdstftp**.

Compile the project with **“Run/Compile”** or **“Run/Clean up and Build”.**



Once the Green bar is displayed it can be transfer to the Ultibo System.

AESEncryptBlock (128bit)
Electronic Codebook (ECB)

AESEncryptBlock (192bit)
Electronic Codebook (ECB)

AESEncryptBlock (256bit)
Electronic Codebook (ECB)

AESDecryptBlock (128bit)
Electronic Codebook (ECB)

AESDecryptBlock (192bit)
Electronic Codebook (ECB)

AESDecryptBlock (256bit)
Electronic Codebook (ECB)

After adding APICrypto.pas

In test_crypto.lpt in

```
var
 AESECBKey:PByte;
 AESECBData:PByte;
 AESECBAESKey:TAESKey;

 AESCBCKey:PByte;
 AESCBCData:PByte;
 AESCBCVector:PByte;

 Cipher:PCipherContext;

 key:String;
 Data:String;
 Actual:String;
 PData:PString;
 Datalen:LongWord;

 InKey:LongWord;
 InKeyStr:String;
 InDataStr:String;
 EncryptDecrypt:LongWord;
```

./upker.sh

Testing 4 blocks
The program test_crypto.lpr now has 2 functions in support of encrption/decryption
Electronic Codebook (ECB)

function
ecbencryption(InKeyStr,InDataStr:String;InKey,EncryptDecrypt:LongWord):String;
Cipher Block Chaining (CBC)
function
cbcencryption(InKeyStr,InDataStr,InIVStr:String;InKey,EncryptDecrypt:LongWord):String;

Steps to encrypt a block of data.

1. Split the data in blocks of 128bits.
   This is what makes up
   Example 16 characters would
   012345678901234567
   'come to dedicte '
   make a block of
   a 128bit block hex when converted from Ascii.
   '636f6d6520746f206465646963746520'

2. Encrypt the first block using a key (128bits, 192bits, or 256bits) using
   the Cipher Block Chaining (CBC) mode and IVector.
   Below are example of (128bits, 192bits, or 256bits)
   128bits
   '2b7e151628aed2a6abf7158809cf4f3c'
   192bits
   '8e73b0f7da0e6452c810f32b809079e562f8ead2522c6b7b'
   256bits
   '603deb1015ca71be2b73aef0857d77811f352c073b6108d72d9810a30914dff4'
   Below is an example IVector
   '000102030405060708090A0B0C0D0E0F'
   The result of the first block will be used as the IVector for the 2nd block.
   With the 256bits as key, the function cbcencryption was used to encryt 2 blocks
   Key '603deb1015ca71be2b73aef0857d77811f352c073b6108d72d9810a30914dff4'
   IVector '000102030405060708090A0B0C0D0E0F'
   Data '636f6d6520746f206465646963746520'

   NewIV for 2nd block '6cafbc0c271b094529e54dd2217dc0'

3. Note: Step3 is optional Decrypt the first block using the same size key to verify that everthing is working okay.
   The same IVector needs to be used.

4. The result of the first block will be used as the IVector for the 2nd block.

In the image below 4 blocks are encrypted/decrypted

256Bit key: 603deb1015ca71be2b73aef0857d77811f352c073b6108d72d9810a30914dff4
Ascii : ‘come to dedicte '
Hex : 636f6d6520746f206465646963746520
Ivector : 00……0F

256Bit key: 603deb1015ca71be2b73aef0857d77811f352c073b6108d72d9810a30914dff4
Ascii : 'a portion of the'
Hex : 6120704f7274696f6e206f6620746865

Ivector : 6c…….c0

256Bit key: 603deb1015ca71be2b73aef0857d77811f352c073b6108d72d9810a30914dff4
Ascii : ' etcided ot emoc'
Hex : 206574636964656420 6f7420656d6f63
Ivector : 46……ac

256Bit key: 603deb1015ca71be2b73aef0857d77811f352c073b6108d72d9810a30914dff4
Ascii : 'eht fo noitrop a'
Hex : 65687420666f206e6f6974724f702061
Ivector : 29……a5

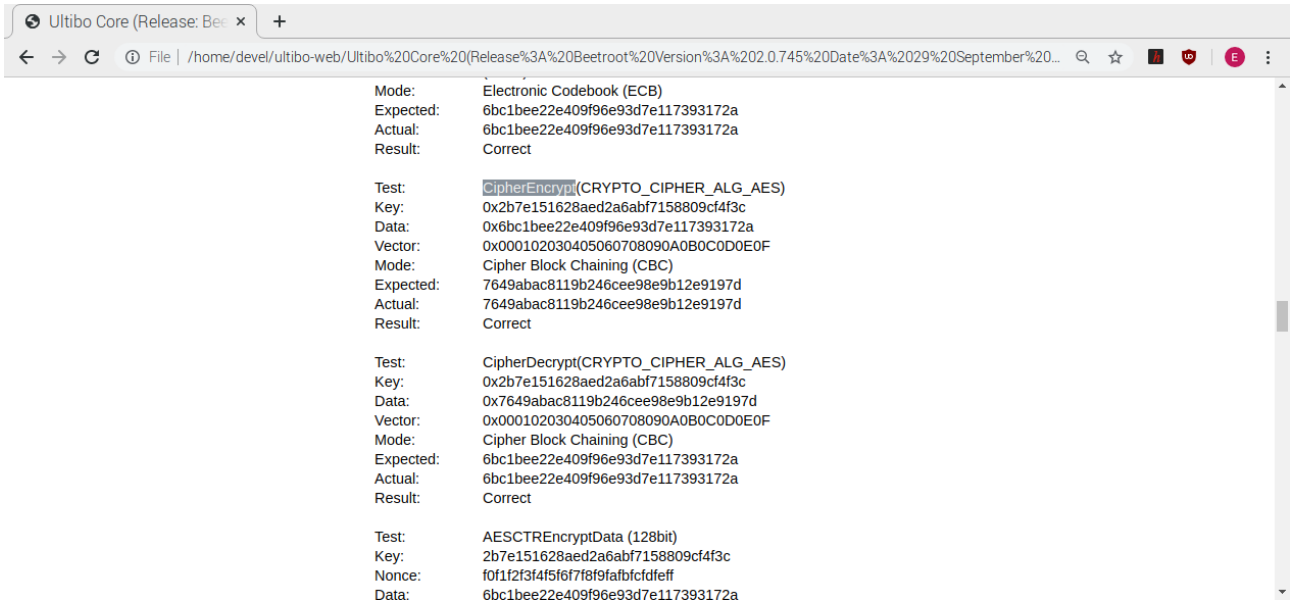
Ultibo Core (Release: Beetroot Version: 2.0.745 Date: 29 September 2019)

```
Starting TFTP_test_crypto example
Local Address 192.168.1.245
TFTP Ready.
first block Ascii come to dedicte
hex of above text 636f6d6520746f206465646963746520

AESEncryptBlock (256bit)
Cipher Block Chaining (CBC)

NewIV will be used as IV of 2nd block 6cafbc0c271bd094539e5e4dd3317dc0
Key:    603deb1015ca71be2b73aef0857d77811f352c073b6108d72d9810a30914dff4
IVector:000102030405060708090A0B0C0D0E0F
Mode:   Cipher Block Chaining (CBC)
Data:   636f6d6520746f206465646963746520
Actual: 6cafbc0c271bd094539e5e4dd3317dc0

AESDecryptBlock (256bit)
Cipher Block Chaining (CBC)
Result: 636f6d6520746f206465646963746520
Key:    603deb1015ca71be2b73aef0857d77811f352c073b6108d72d9810a30914dff4
IVector:000102030405060708090A0B0C0D0E0F
Mode:   Cipher Block Chaining (CBC)
Data:   6cafbc0c271bd094539e5e4dd3317dc0
Actual: 636f6d6520746f206465646963746520
NewIV 6cafbc0c271bd094539e5e4dd3317dc0
S1 6cafbc0c271bd094539e5e4dd3317dc0

2nd  block Ascii a portion of the
hex of above text 6120704f7274696f6e206f6620746865

AESEncryptBlock (256bit)
Cipher Block Chaining (CBC)
Key:    603deb1015ca71be2b73aef0857d77811f352c073b6108d72d9810a30914dff4
IVector:6cafbc0c271bd094539e5e4dd3317dc0
Mode:   Cipher Block Chaining (CBC)
Data:   6120704f7274696f6e206f6620746865
Actual: 46f299c63a2ea6e49bad0c81f39a55ac
S2 46f299c63a2ea6e49bad0c81f39a55ac

AESDecryptBlock (256bit)
Cipher Block Chaining (CBC)
Key:    603deb1015ca71be2b73aef0857d77811f352c073b6108d72d9810a30914dff4
IVector:6cafbc0c271bd094539e5e4dd3317dc0
Mode:   Cipher Block Chaining (CBC)
Data;   46f299c63a2ea6e49bad0c81f39a55ac
Actual: 6120704f7274696f6e206f6620746865
```

```
third block Ascii  etcided ot emoc
hex of above text 206574636964656420f7420656d6f63

AESEncryptBlock (256bit)
Cipher Block Chaining (CBC)

S2 will be used as IV of 3rd block 46f299c63a2ea6e49bad0c81f39a55ac
Key:    603deb1015ca71be2b73aef0857d77811f352c073b6108d72d9810a30914dff4
IVector:46f299c63a2ea6e49bad0c81f39a55ac
Mode:   Cipher Block Chaining (CBC)
Data:   206574636964656420f7420656d6f63
Actual: 29261323683144c05574d0545c8bbfa5

AESDecryptBlock (256bit)
Cipher Block Chaining (CBC)
Result: 206574636964656420f7420656d6f63
Key:    603deb1015ca71be2b73aef0857d77811f352c073b6108d72d9810a30914dff4
IVector:46f299c63a2ea6e49bad0c81f39a55ac
Mode:   Cipher Block Chaining (CBC)
Data:   29261323683144c05574d0545c8bbfa5
Actual: 206574636964656420f7420656d6f63

NewIV 29261323683144c05574d0545c8bbfa5
S1 29261323683144c05574d0545c8bbfa5

4th  block Ascii eht fo noitrop a
hex of above text 65687420666f206e6f6974724f702061

AESEncryptBlock (256bit)
Cipher Block Chaining (CBC)
Key:    603deb1015ca71be2b73aef0857d77811f352c073b6108d72d9810a30914dff4
IVector:29261323683144c05574d0545c8bbfa5
Mode:   Cipher Block Chaining (CBC)
Data:   65687420666f206e6f6974724f702061
Actual: 00c70caf677d5a9b26367a641eb19ad9
S2 00c70caf677d5a9b26367a641eb19ad9

AESDecryptBlock (256bit)
Cipher Block Chaining (CBC)
Key:    603deb1015ca71be2b73aef0857d77811f352c073b6108d72d9810a30914dff4
IVector:29261323683144c05574d0545c8bbfa5
Mode:   Cipher Block Chaining (CBC)
Data:   00c70caf677d5a9b26367a641eb19ad9
Actual: 65687420666f206e6f6974724f702061
```

Testing of ECB & CBC

```
Ultibo Core (Release: Beetroot Version: 2.0.745 Date: 29 September 2019)

Starting TFTP_test_crypto example
Local Address 192.168.1.245
TFTP Ready.

AESEncryptBlock (128bit)
Electronic Codebook (ECB)
Key:    2b7e151628aed2a6abf7158809cf4f3c
Data:   6bc1bee22e409f96e93d7e117393172a
Actual: 3ad77bb40d7a3660a89ecaf32466ef97

AESDecryptBlock (128bit)
Electronic Codebook (ECB)
Key:    2b7e151628aed2a6abf7158809cf4f3c
Data:   3ad77bb40d7a3660a89ecaf32466ef97
Actual: 6bc1bee22e409f96e93d7e117393172a

AESEncryptBlock (192bit)
Electronic Codebook (ECB)
Key:    8e73b0f7da0e6452c810f32b809079e562f8ead2522c6b7b
Data:   6bc1bee22e409f96e93d7e117393172a
Actual: bd334f1d6e45f25ff712a214571fa5cc

AESDecryptBlock (192bit)
Electronic Codebook (ECB)
Key:    8e73b0f7da0e6452c810f32b809079e562f8ead2522c6b7b
Data:   bd334f1d6e45f25ff712a214571fa5cc
Actual: 6bc1bee22e409f96e93d7e117393172a

AESEncryptBlock (256bit)
Electronic Codebook (ECB)
Key:    603deb1015ca71be2b73aef0857d77811f352c073b6108d72d9810a30914dff4
Data:   6bc1bee22e409f96e93d7e117393172a
Actual: f3eed1bdb5d2a03c064b5a7e3db181f8

AESDecryptBlock (256bit)
Electronic Codebook (ECB)
Key:    603deb1015ca71be2b73aef0857d77811f352c073b6108d72d9810a30914dff4
Data:   f3eed1bdb5d2a03c064b5a7e3db181f8
Actual: 6bc1bee22e409f96e93d7e117393172a
Transfer for "kernel7.img" started.
Transfer for kernel7.img complete.

AESEncryptBlock (128bit)
Cipher Block Chaining (CBC)
Key:    2b7e151628aed2a6abf7158809cf4f3c
IVector:000102030405060708090A0B0C0D0E0F
Mode:   Cipher Block Chaining (CBC)
Data:   6bc1bee22e409f96e93d7e117393172a
Actual: 7649abac8119b246cee98e9b12e9197d

AESDecryptBlock (128bit)
Cipher Block Chaining (CBC)
Key:    2b7e151628aed2a6abf7158809cf4f3c
IVector:000102030405060708090A0B0C0D0E0F
Mode:   Cipher Block Chaining (CBC)
Data:   7649abac8119b246cee98e9b12e9197d
Actual: 6bc1bee22e409f96e93d7e117393172a

AESEncryptBlock (192bit)
Cipher Block Chaining (CBC)
Key:    8e73b0f7da0e6452c810f32b809079e562f8ead2522c6b7b
IVector:000102030405060708090A0B0C0D0E0F
Mode:   Cipher Block Chaining (CBC)
Data:   6bc1bee22e409f96e93d7e117393172a
Actual: 4f021db243bc633d7178183a9fa071e8

AESDecryptBlock (192bit)
Cipher Block Chaining (CBC)
Key:    8e73b0f7da0e6452c810f32b809079e562f8ead2522c6b7b
IVector:000102030405060708090A0B0C0D0E0F
Mode:   Cipher Block Chaining (CBC)
Data:   4f021db243bc633d7178183a9fa071e8
Actual: 6bc1bee22e409f96e93d7e117393172a

AESEncryptBlock (256bit)
Cipher Block Chaining (CBC)
Key:    603deb1015ca71be2b73aef0857d77811f352c073b6108d72d9810a30914dff4
IVector:000102030405060708090A0B0C0D0E0F
Mode:   Cipher Block Chaining (CBC)
Data:   6bc1bee22e409f96e93d7e117393172a
Actual: f58c4c04d6e5f1ba779eabfb5f7bfbd6

AESDecryptBlock (256bit)
Cipher Block Chaining (CBC)
Key:    603deb1015ca71be2b73aef0857d77811f352c073b6108d72d9810a30914dff4
IVector:000102030405060708090A0B0C0D0E0F
Mode:   Cipher Block Chaining (CBC)
Data:   f58c4c04d6e5f1ba779eabfb5f7bfbd6
Actual: 6bc1bee22e409f96e93d7e117393172a
```



Now the results match the results on http://192.168.1.245/status/cryptoapi/

(219 unread) - develone | ultibo.org - New posts | Ultibo Core (Release: Bee

File | /home/devel/ultibo-web/Ultibo%20Core%20(Release%3A%20Beetroot%20Version%3A%202.0.745%20Date%3A...

AES Cipher Tests

Test:	AESEncryptBlock (128bit)
Key:	0x2b7e151628aed2a6abf7158809cf4f3c
Data:	0x6bc1bee22e409f96e93d7e117393172a
Vector:	(None)
Mode:	Electronic Codebook (ECB)
Expected:	3ad77bb40d7a3660a89ecaf32466ef97
Actual:	3ad77bb40d7a3660a89ecaf32466ef97
Result:	Correct

Test:	AESEncryptBlock (192bit)
Key:	0x8e73b0f7da0e6452c810f32b809079e562f8ead2522c6b7b
Data:	0x6bc1bee22e409f96e93d7e117393172a
Vector:	(None)
Mode:	Electronic Codebook (ECB)
Expected:	bd334f1d6e45f25ff712a214571fa5cc
Actual:	bd334f1d6e45f25ff712a214571fa5cc
Result:	Correct

Test:	AESEncryptBlock (256bit)
Key:	0x603deb1015ca71be2b73aef0857d77811f352c073b6108d72d9810a30914dff4
Data:	0x6bc1bee22e409f96e93d7e117393172a
Vector:	(None)
Mode:	Electronic Codebook (ECB)
Expected:	f3eed1bdb5d2a03c064b5a7e3db181f8

(219 unread) - develone | Ultibo Core (Release: Bee

File | /home/devel/ultibo-web/Ultibo%20Core%20(Release%3A%20Beetroot%20Version%3A%202.0.745%20Date%3A...

Test:	AESDecryptBlock (128bit)
Key:	0x2b7e151628aed2a6abf7158809cf4f3c
Data:	0x3ad77bb40d7a3660a89ecaf32466ef97
Vector:	(None)
Mode:	Electronic Codebook (ECB)
Expected:	6bc1bee22e409f96e93d7e117393172a
Actual:	6bc1bee22e409f96e93d7e117393172a
Result:	Correct

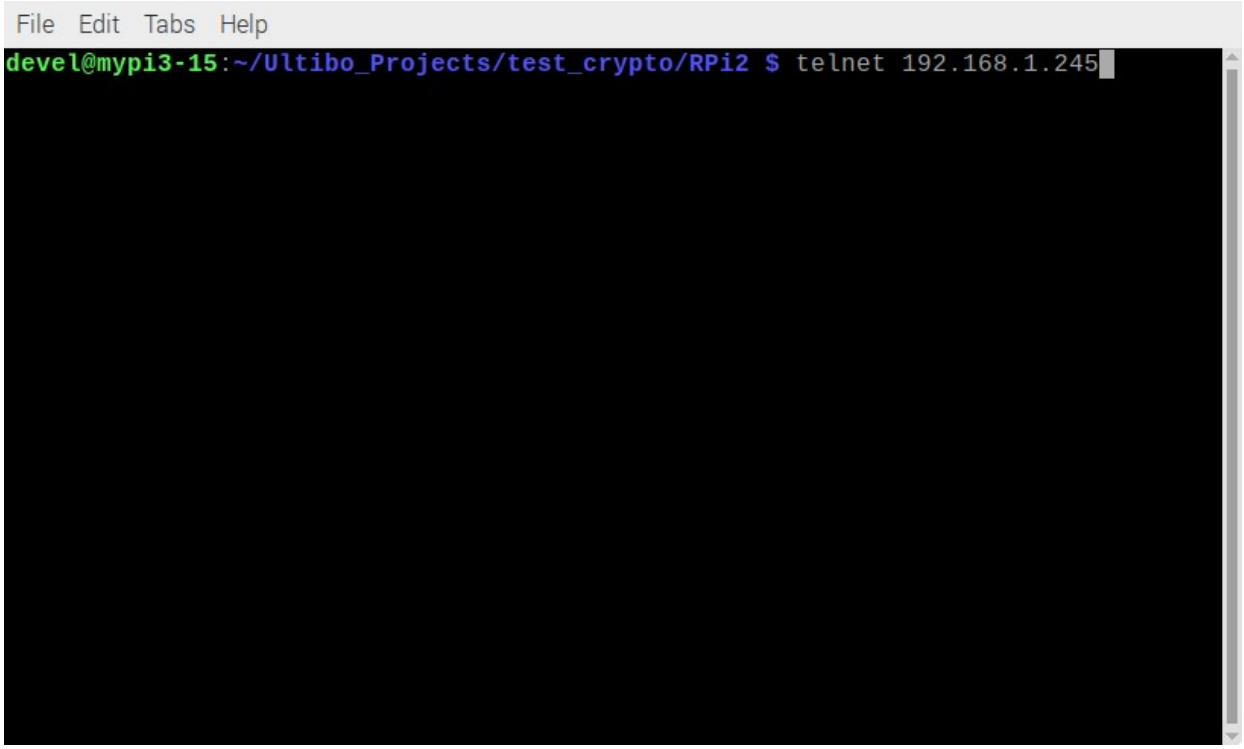
Test:	AESDecryptBlock (192bit)
Key:	0x8e73b0f7da0e6452c810f32b809079e562f8ead2522c6b7b
Data:	0xbd334f1d6e45f25ff712a214571fa5cc
Vector:	(None)
Mode:	Electronic Codebook (ECB)
Expected:	6bc1bee22e409f96e93d7e117393172a
Actual:	6bc1bee22e409f96e93d7e117393172a
Result:	Correct

Test:	AESDecryptBlock (256bit)
Key:	0x603deb1015ca71be2b73aef0857d77811f352c073b6108d72d9810a30914dff4
Data:	0xf3eed1bdb5d2a03c064b5a7e3db181f8
Vector:	(None)
Mode:	Electronic Codebook (ECB)
Expected:	6bc1bee22e409f96e93d7e117393172a

encrdecr.jpeg | Show all

Decryption APICrypto.pas

shell1



shell2

```
File  Edit  Tabs  Help
29-3-20 02:24:18          3798568  start_x.elf
29-3-20 02:24:18          3145850  t
29-3-20 02:24:20           635016  teapot.obj.dat
29-3-20 02:23:56               24  testfile
29-3-20 02:24:20         27983872  test.h264
29-3-20 02:24:24              500  test.html
10-4-20 16:23:58             7848  test.j2k
6-4-20 17:37:26            196730  test_wr.bmp
29-3-20 02:24:24             1718  ultibologging.log
29-3-20 02:24:24         27983872  v1.h264
29-3-20 02:24:30          1002763  v2.h264
29-3-20 02:24:30      <DIR>        www
2-4-20 17:31:26             65596  red.pgm
2-4-20 17:31:38             65596  grn.pgm
2-4-20 17:31:52             65596  blu.pgm
6-4-20 11:23:30              1024  Sred.bin
6-4-20 11:23:34              1024  Sgrn.bin
6-4-20 11:23:36            262144  rcgrn.bin
6-4-20 11:23:38              1024  Sblu.bin
6-4-20 11:23:38            262144  rcblu.bin
         69 file(s) 136527430 bytes
         2 dir(s)

C:\>
```

Webstatus

(211 unread) - deve × | Wifi - Page 2 - ultib × | Common Vulnerabi × | develone/tiny-AES1 × | Ultibo Core (Releas × | +

Not secure | 192.168.1.245/status

Ultibo Core (Release: Beetroot Version: 2.0.745 Date: 29 September 2019)

General		**General**
Platform		
Memory	Release Name:	Beetroot
Heap Blocks	Release Version:	2.0.745
CPU	Release Date:	29 September 2019
FPU		
GPU	Time (Local):	30-12-99 00:00:08
RTL	Time (UTC):	30-12-99 00:00:08
Clock		
Locale	Timezone:	UTC
Threading		
Thread List	Daylight Start:	None
Scheduler	Daylight Date:	N/A
Devices		
Drivers	Standard Start:	None
Handles	Standard Date:	N/A
USB		
MMC / SD	Uptime:	0 days 00:00:08
Network		
Storage		
Filesystem		
Disk Cache		
Keyboard		
Mouse		
Framebuffer		