# AES Encryption-Rijndael Block Cipher to Symbian OS

This code snippet provides an implementation of the AES Encryption-Rijndael Block Cipher on Symbian. The implementation can be used for encrypting/decrypting on Symbian, but if you want to decrypt on another platform you should encrypt using the Symbian Cryptography APIs .

## Rijndael Symbian OS Interface

```
#include "rijndael-alg-symbianos.h"


namespace aes {


/* Modes */
#define  DIR_ENCRYPT  0  // Encrypting?
#define  DIR_DECRYPT  1  // Decrypting?
#define  MODE_ECB     1  // Ciphering in ECB?
#define  MODE_CBC     2  // Ciphering in CBC?
#define  MODE_CFB1    3  // Ciphering in 1-bit CFB?


/* Block Size */
#define  BITSPERBLOCK 128  // Bits in a cipher block


/* Error Codes */
#define  BAD_KEY_DIR  -1  // Key direction is invalid
#define  BAD_KEY_MAT  -2  // Key not of correct length
#define  BAD_KEY_INSTANCE -3  // Key is not valid
#define  BAD_CIPHER_MODE  -4  // Cipher instance invalid
#define  BAD_CIPHER_STATE -5  // Cipher in wrong state
#define  BAD_CIPHER_INSTANCE -7


/* Keys */
#define  MAX_KEY_SIZE 64  // # of ASCII char's of a key
#define  MAX_IV_SIZE  BITSPERBLOCK/8  // # bytes of an IV


/* The structure for key information */
typedef struct {
  TUint8  direction;    // Encrypting or decrypting?
  TInt    keyLen;       // Length of the key
  TText8  keyMaterial[MAX_KEY_SIZE+1];  // Raw key
  TInt    blockLen;     // Block Length
  word8   keySched[MAXROUNDS+1][4][MAXBC];
} keyInstance;


/* The structure for cipher information */
typedef struct {
  TUint8  mode;    // MODE_ECB, MODE_CBC, or MODE_CFB1
  TUint8  IV[MAX_IV_SIZE]; // Initialization vector
  TInt    blockLen;
} cipherInstance;


/* ********* */
/* Functions */
/* ********* */

TInt makeKey(keyInstance *key,
             TUint8 direction,
             TInt keyLen,
             const TText8 *keyMaterial);

TInt cipherInit(cipherInstance *cipher,
                TUint8 mode,
                const TText8 *IV);

TInt blockEncrypt(const cipherInstance *cipher,
                  keyInstance *key,
                  TUint8 *input,
                  TInt inputLen,
                  TUint8 *outBuffer);

TInt blockDecrypt(const cipherInstance *cipher,
                  keyInstance *key,
                  TUint8 *input,
                  TInt inputLen,
                  TUint8 *outBuffer);

TInt cipherUpdateRounds(const cipherInstance *cipher,
                        keyInstance *key,
                        TUint8 *input,
                        TInt inputLen,
                        TUint8 *outBuffer,
                        TInt Rounds);

} // namespace aes
```

## Working with Rijndael

```
// Secret message (which fits exactly in one block)
_LIT8(KMySecret, "Hello world!!!!!");
TBuf8<128> secret(KMySecret);
```


(C) Copyright Nokia 2013. All rights reserved.

```
//
// Init stuff
TUint8 block[4*MAXBC];
TPtr8 blockPtr(block, BITSPERBLOCK/8);
aes::keyInstance keyInst;
aes::cipherInstance cipherInst;


//
// Generate key instance for encryption.
// The secret key material is an hex string
// with a length of keyLen/4 bytes.
TInt keyLen = 128;
TText8* secretKeyMaterial =
         (TText8*) "E5E6E7E9EA292A2B2D256789012345E5";
keyInst.blockLen = BITSPERBLOCK;
aes::makeKey(&keyInst, DIR_ENCRYPT,
             keyLen, secretKeyMaterial);


//
// Init cipher instance
cipherInst.blockLen = BITSPERBLOCK;
aes::cipherInit(&cipherInst, MODE_ECB, NULL);


//
// Encrypt block
// Fill block with our secret message
blockPtr.Append(secret.Ptr(), BITSPERBLOCK/8);
aes::blockEncrypt(&cipherInst, &keyInst,
                  block, BITSPERBLOCK, block);
TBuf8<BITSPERBLOCK/8> cipherText;
cipherText.Append(blockPtr);
blockPtr.Zero();


//
// cipherText contains the encrypted secret


//
// Generate key instance for decryption
keyInst.blockLen = BITSPERBLOCK;
aes::makeKey(&keyInst, DIR_DECRYPT,
             keyLen, secretKeyMaterial);


//
// Init cipher instance
cipherInst.blockLen = BITSPERBLOCK;
aes::cipherInit(&cipherInst, MODE_ECB, NULL);


//
// Decrypt block
blockPtr.Append(cipherText.Ptr(), BITSPERBLOCK/8);
aes::blockDecrypt(&cipherInst, &keyInst,
                  block, BITSPERBLOCK, block);
TBuf8<BITSPERBLOCK/8> decryptedText;
decryptedText.Append(blockPtr);


//
// decryptedText contains "Hello world!!!!!"
```


(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.