1. Contents for Encryption

Mvd\_sign, Mvd\_level\_suffix, Coef\_sign, Coef\_level\_suffix

1. Process of encryption

First of all, generate an initial vector(IV) and AES key, 128 bits of each, and the default vale is

‘0000000000000000’ and ‘0123456789ABCDEF’;

The IV is used for XORing with plain text, and the result will be encrypted with the AES key as the new IV for the next loop.

The decryption process is the same as the encryption.

1. Class for Encryption

class EncAndDec //used for encryption and decryption

{

public:

EncAndDec();

~EncAndDec();

Void Creat(Int enc); //to initial the member of the class

Bool getBitKey(UInt &key); // generate a key of one bit

Void GenerateEncIV(); // to generate the initial vector for encryption

Void EncOneBit(UInt &bit); // encrypt one bit

Void GenerateDecIV(); //generate the initial vector for decryption

Void DecOneBit(UInt &bit); //decrypt one bit

Void Reset();

private:

Bool m\_fisrtset;

Int m\_operate; //the variable equal to 1 stands for encryption while 0

//stands for decryption

AES\_KEY m\_userEncKey; //to store userkey

AES\_KEY m\_userDecKey;

UChar m\_userKey[AES\_BLOCK\_SIZE]; //for inputing 128 bits key

UChar m\_initialIV[AES\_BLOCK\_SIZE];

UChar m\_IV[AES\_BLOCK\_SIZE]; // initial vector

UChar m\_EIV[AES\_BLOCK\_SIZE]; // to store the result of plain text XOR with IV

// and will be encrypted with AES function

UInt m\_BitNum; //calculate the number of encrypted or

// decrypted bits

};