#include <gcrypt.h>

int main()

{

#define GCRY\_CIPHER GCRY\_CIPHER\_AES128 // Pick the cipher here

#define GCRY\_C\_MODE GCRY\_CIPHER\_MODE\_ECB // Pick the cipher mode here

gcry\_error\_t gcryError;

gcry\_cipher\_hd\_t gcryCipherHd;

size\_t index;

size\_t keyLength = gcry\_cipher\_get\_algo\_keylen(GCRY\_CIPHER);

size\_t blkLength = gcry\_cipher\_get\_algo\_blklen(GCRY\_CIPHER);

char \* txtBuffer = "123456789 abcdefghijklmnopqrstuvwzyz ABCDEFGHIJKLMNOPQRSTUVWZYZ";

size\_t txtLength = strlen(txtBuffer)+1; // string plus termination

char \* encBuffer = malloc(txtLength);

char \* outBuffer = malloc(txtLength);

char \* aesSymKey = "one test AES key"; // 16 bytes

char \* iniVector = "a test ini value"; // 16 bytes

/\*Make sure important subsystems are initialized\*/

if (!gcry\_check\_version (GCRYPT\_VERSION))

{

fputs("libgcrypt version mismatch\n", stderr);

exit(2);

}

/\* We don't care about secure memory for this example \*/

gcry\_control(GCRYCTL\_DISABLE\_SECMEM, 0);

/\* We're done with initialization \*/

gcry\_control(GCRYCTL\_INITIALIZATION\_FINISHED, 0);

gcryError = gcry\_cipher\_open(

&gcryCipherHd, // gcry\_cipher\_hd\_t \*

GCRY\_CIPHER, // int

GCRY\_C\_MODE, // int

0); // unsigned int

if (gcryError)

{

printf("gcry\_cipher\_open failed: %s/%s\n",

gcry\_strsource(gcryError),

gcry\_strerror(gcryError));

return;

}

printf("gcry\_cipher\_open worked\n");

gcryError = gcry\_cipher\_setkey(gcryCipherHd, aesSymKey, keyLength);

if (gcryError)

{

printf("gcry\_cipher\_setkey failed: %s/%s\n",

gcry\_strsource(gcryError),

gcry\_strerror(gcryError));

return;

}

printf("gcry\_cipher\_setkey worked\n");

gcryError = gcry\_cipher\_setiv(gcryCipherHd, iniVector, blkLength);

if (gcryError)

{

printf("gcry\_cipher\_setiv failed: %s/%s\n",

gcry\_strsource(gcryError),

gcry\_strerror(gcryError));

return;

}

printf("gcry\_cipher\_setiv worked\n");

gcryError = gcry\_cipher\_encrypt(

gcryCipherHd, // gcry\_cipher\_hd\_t

encBuffer, // void \*

txtLength, // size\_t

txtBuffer, // const void \*

txtLength); // size\_t

if (gcryError)

{

printf("gcry\_cipher\_encrypt failed: %s/%s\n",

gcry\_strsource(gcryError),

gcry\_strerror(gcryError));

return;

}

printf("gcry\_cipher\_encrypt worked\n");

gcryError = gcry\_cipher\_setiv(gcryCipherHd, iniVector, blkLength);

if (gcryError)

{

printf("gcry\_cipher\_setiv failed: %s/%s\n",

gcry\_strsource(gcryError),

gcry\_strerror(gcryError));

return;

}

printf("gcry\_cipher\_setiv worked\n");

gcryError = gcry\_cipher\_decrypt(

gcryCipherHd, // gcry\_cipher\_hd\_t

outBuffer, // void \*

txtLength, // size\_t

encBuffer, // const void \*

txtLength); // size\_t

if (gcryError)

{

printf("gcry\_cipher\_decrypt failed: %s/%s\n",

gcry\_strsource(gcryError),

gcry\_strerror(gcryError));

return;

}

printf("gcry\_cipher\_decrypt worked\n");

printf("keyLength = %zu\n", keyLength);

printf("blkLength = %zu\n", blkLength);

printf("txtLength = %zu\n", txtLength);

printf("aesSymKey = %s\n", aesSymKey);

printf("iniVector = %s\n", iniVector);

printf("txtBuffer = %s\n", txtBuffer);

printf("encBuffer = ");

for (index = 0; index<txtLength; index++)

printf("%02X", (unsigned char)encBuffer[index]);

printf("\n");

printf("outBuffer = %s\n", outBuffer);

// close cipher and clean up memory

gcry\_cipher\_close(gcryCipherHd);

free(encBuffer);

free(outBuffer);

}