**AES ALGORITHM IN CRYPTOGRAPHY ALGORITHM IMPLEMENT IN C**

|  |
| --- |
| #include<stdio.h> |
|  | #include <stdlib.h> |
|  | #include <string.h> |
|  |  |
|  | /\* |
|  | \* MCrypt API available online: |
|  | \* http://linux.die.net/man/3/mcrypt |
|  | \*/ |
|  | #include <mcrypt.h> |
|  |  |
|  | #include <math.h> |
|  | #include <stdint.h> |
|  | #include <stdlib.h> |
|  |  |
|  | int encrypt( |
|  | void\* buffer, |
|  | int buffer\_len, /\* Because the plaintext could include null bytes\*/ |
|  | char\* IV, |
|  | char\* key, |
|  | int key\_len |
|  | ){ |
|  | MCRYPT td = mcrypt\_module\_open("rijndael-128", NULL, "cbc", NULL); |
|  | int blocksize = mcrypt\_enc\_get\_block\_size(td); |
|  | if( buffer\_len % blocksize != 0 ){return 1;} |
|  |  |
|  | mcrypt\_generic\_init(td, key, key\_len, IV); |
|  | mcrypt\_generic(td, buffer, buffer\_len); |
|  | mcrypt\_generic\_deinit (td); |
|  | mcrypt\_module\_close(td); |
|  |  |
|  | return 0; |
|  | } |
|  |  |
|  | int decrypt( |
|  | void\* buffer, |
|  | int buffer\_len, |
|  | char\* IV, |
|  | char\* key, |
|  | int key\_len |
|  | ){ |
|  | MCRYPT td = mcrypt\_module\_open("rijndael-128", NULL, "cbc", NULL); |
|  | int blocksize = mcrypt\_enc\_get\_block\_size(td); |
|  | if( buffer\_len % blocksize != 0 ){return 1;} |
|  |  |
|  | mcrypt\_generic\_init(td, key, key\_len, IV); |
|  | mdecrypt\_generic(td, buffer, buffer\_len); |
|  | mcrypt\_generic\_deinit (td); |
|  | mcrypt\_module\_close(td); |
|  |  |
|  | return 0; |
|  | } |
|  |  |
|  | void display(char\* ciphertext, int len){ |
|  | int v; |
|  | for (v=0; v<len; v++){ |
|  | printf("%d ", ciphertext[v]); |
|  | } |
|  | printf("\n"); |
|  | } |
|  |  |
|  | int main() |
|  | { |
|  | MCRYPT td, td2; |
|  | char\* plaintext = "test text 123"; |
|  | char\* IV = "AAAAAAAAAAAAAAAA"; |
|  | char\* key = "0123456789abcdef"; |
|  | int keysize = 16; /\* 128 bits \*/ |
|  | char\* buffer; |
|  | int buffer\_len = 16; |
|  |  |
|  | buffer = calloc(1, buffer\_len); |
|  | strncpy(buffer, plaintext, buffer\_len); |
|  |  |
|  | printf("==C==\n"); |
|  | printf("plain: %s\n", plaintext); |
|  | encrypt(buffer, buffer\_len, IV, key, keysize); |
|  | printf("cipher: "); display(buffer , buffer\_len); |
|  | decrypt(buffer, buffer\_len, IV, key, keysize); |
|  | printf("decrypt: %s\n", buffer); |
|  |  |
|  | return 0; |
|  | } |

