Program 31 C program for subkey generation in CMAC

#include <stdio.h>

typedef unsigned char byte;

void print\_hex(byte \*data, int length) {

for (int i = 0; i < length; i++) {

printf("%02x", data[i]);

}

printf("\n");

}

void generate\_subkeys(byte \*initial\_key, int block\_size, byte \*subkey1, byte \*subkey2) {

byte L[block\_size / 8];

byte const\_Rb[block\_size / 8];

byte zero[block\_size / 8] = {0};

// Calculate L

byte msb = (initial\_key[0] & 0x80) ? 0x87 : 0x00;

for (int i = 0; i < block\_size / 8; i++) {

L[i] = (initial\_key[i] << 1) | ((i < block\_size / 8 - 1) ? (initial\_key[i + 1] >> 7) : 0);

subkey1[i] = L[i] ^ const\_Rb[i];

}

// Left shift L

byte carry = (L[0] & 0x80) ? 1 : 0;

for (int i = 0; i < block\_size / 8; i++) {

L[i] = (L[i] << 1) | carry;

carry = (L[i] & 0x80) ? 1 : 0;

}

// Calculate subkey2

for (int i = 0; i < block\_size / 8; i++) {

subkey2[i] = L[i] ^ const\_Rb[i];

}

}

int main() {

// Set block size (64 or 128 bits)

int block\_size = 128;

byte initial\_key[block\_size / 8];

byte subkey1[block\_size / 8];

byte subkey2[block\_size / 8];

// Initialize initial\_key with your key data here

// ...

// Calculate constants based on block size

byte const\_Rb[block\_size / 8];

if (block\_size == 64) {

byte const\_64 = 0x1B; // Rb for 64-bit block size

for (int i = 0; i < block\_size / 8; i++) {

const\_Rb[i] = const\_64;

}

} else if (block\_size == 128) {

byte const\_128 = 0x87; // Rb for 128-bit block size

for (int i = 0; i < block\_size / 8; i++) {

const\_Rb[i] = const\_128;

}

} else {

printf("Unsupported block size\n");

return 1;

}

generate\_subkeys(initial\_key, block\_size, subkey1, subkey2);

printf("Subkey 1: ");

print\_hex(subkey1, block\_size / 8);

printf("Subkey 2: ");

print\_hex(subkey2, block\_size / 8);

return 0;

}

OUTPUT:

Subkey 1: 80fafd0000000000efec8a0000000000

Subkey 2: 61f35500000000004ce3d50100000000