CS 2160 Fall 2017 Al Glock

Program 3

Write code to encrypt and decrypt a message using symmetric encryption. There are three methods you’ll need to write for this assignment. The first is a hash method. A key value is supplied, and in successive calls to the hash method, you will produce different results which will be used to encrypt and decrypt the message text.

The Hash method takes two arguments and produces one value. $a0 contains the value of the previous hash (will be 0 the first time) and $a1 contains the key. The new hash value is created by multiplying the previous hash value by 31 and then adding the key value to it. This result is returned in $v0. For the next call to Hash, you’ll need to move the output value in $v0 to the input value in $a0. The first five calls to Hash should take in and produce these values:

|  |  |  |
| --- | --- | --- |
| **$a0** | **$a1** | **$v0** |
| 0 | 0x1234abcd | 0x1234abcd |
| 0x1234abcd | 0x469579a0 | 0x469579a0 |
| 0x469579a0 | 0x9e4e662d | 0x9e4e662d |
| 0x9e4e662d | 0x3db30b40 | 0x3db30b40 |
| 0x3db30b40 | 0x8ae3088d | 0x8ae3088d |

The Encrypt method operates by reading a byte from the message text, calling the Hash method and XORing the hash result with the message byte and storing the low-order byte of the result in the cipher text buffer. The Encrypt method accepts an input text buffer address in $a0 and the output buffer in $a1. The hash method is initialized with the Key and since calling arguments are not (supposed to be) changed, it will remain the same throughout the Encrypt application.

The Decrypt method “undoes” the encryption formerly applied. However, the method is identical. You would read the cipher text byte by byte, call the hash method each tine, XOR the hash result with the encrypted byte. Then write that byte to the plaintext buffer. The address of the ciphertext buffer is supplied in $a0 and the address of the plaintext buffer is in $a1. There is no return value.

The output you receive should look like: this:

Original message text:

This string will be encrypted.

Encrypted text (hex):

0x3344c899

0xf29993ad

0xe0ca4e24

0x6c01097a

0x6048c2ed

0xf28e8ee8

0xa5d95034

0x006d4e69

0x00000000

0x00000000

Decrypted text:

This string will be encrypted.

-- program is finished running --

If your hash method produces the values shown above, then it works. Your encryption method should produce the results shown in the output above. Check the hex code to make sure you’re getting the proper values before continuing on to the Decrypt method.

Add your code at the very bottom of the code below. Do not change any of the code above that point in your submission. Altering the boilerplate code will result in a 0 on the assignment because it potentially changes the problem you’re trying to solve.

The boilerplate code is commented, so you are encouraged to study it to see how it works. This may help you when you add in your code. Basically, the number 100 is loaded into the $a0 register and a separate routine is called to compute the sum of the numbers. Its result is saved in register $s0. Notice that this routine does not alter the calling argument in $a0. Your summation routine is called next and its result is compared with the previously stored result. If the two results compare, you get the success message. If not, you don’t. You should call your methods Encrypt and Decrypt. If you don’t, the boilerplate code can’t call your method. And of course, don’t change the boilerplate code.

You will lose points on this assignment if you 1) submit code you didn’t write yourself, 2) alter the boilerplate code in the file you turn in, 3) name the file something besides <last name>.asm, 4) don’t get the right answer, 5) alter the values in $a0 or $s0, 6) do not return properly from your subroutine, 7) have side effects in your code (i.e., dependencies on other code apart from $a and $v register values), and 8) can’t explain your code if I ask you about it.

.data

Key:

.word 0x1234ABCD

Messagetext:

.asciiz "This string will be encrypted."

Output1:

.asciiz "Original message text:\r\n"

Output2:

.asciiz "\r\nEncrypted text (hex):\r\n"

Output3:

.asciiz "\r\nDecrypted text:\r\n"

CR:

.asciiz "\r\n"

.align 2

Ciphertext:

.space 40

.align 2

Plaintext:

.space 40

.text #most of this code is string printing

la $a0, Output1

addi $v0, $0, 4

syscall

la $a0, Messagetext

addi $v0, $0, 4

syscall

la $a0, CR

addi $v0, $0, 4

syscall

la $a0, Messagetext

la $a1, Ciphertext

jal Encrypt #encrypt the message text to the ciphertext buffer

la $a0, Output2

addi $v0, $0, 4

syscall

la $a0, Ciphertext

jal PrintBufferHex #print out the hex of the ciphertext

la $a0, CR

addi $v0, $0, 4

syscall

addi $a0, $a1, 0

la $a1, Plaintext

jal Decrypt #decrypt the cipher text to plaintext

la $a0, Output3

addi $v0, $0, 4

syscall

addi $a0, $a1, 0

addi $v0, $0, 4

syscall

la $a0, CR

addi $v0, $0, 4

syscall

addi $v0, $0, 10 #end of program

syscall

PrintBufferHex: #subroutine which prints buffer in hex

addi $sp, $sp, -4

sw $a0, ($sp)

addi $t0, $a0, 0

addi $t1, $0, 10

PrintLoop:

addi $v0, $0, 34

lw $a0, ($t0)

syscall

la $a0, CR

addi $v0, $0, 4

syscall

addi $t0, $t0, 4

addi $t1, $t1, -1

bne $t1, $0, PrintLoop

lw $a0, ($sp)

addi $sp, $sp, 4

jr $ra

Encrypt:

#your code goes here