CS 401

Term Project Phase III

Mustafa Onuralp Devres 29310

Ege Kaan Özalp 28989

4 Encryption

In this phase given an encrypted plaintext P, the given message is encrypted and saved to the ciphertext C.

```
IV: .word 0x3412, 0x7856, 0xBC9A, 0xF0DE
K: .word 0x2301, 0x6745, 0xAB89, 0xEFCD, 0xDCFE, 0x98BA, 0x5476, 0x1032
R: .space 32 #Space for R0 to R7
P: .word 0x1100, 0x3322, 0x5544, 0x7766, 0x9988, 0xBBAA, 0xDDCC, 0xFFEE
C: .space 32 #Space for C0 to C7
```

Figure 1. The Vectors

The Plaintext input is given from the data section and there is a blank space created for the storage of the encrypted text.

```
encryption:

    addi $sp, $sp, -36
    sw $s3, 0($sp)
    sw $s4, 4($sp)
    sw $s5, 8($sp)
    sw $s0, 12($sp)
    sw $s1, 16($sp)
    sw $s2, 20($sp)
    sw $s6, 24($sp)
    sw $s7, 28($sp)
    sw $ra, 32($sp)


    #Load addresses
    la $s3, R
    la $s4, K
    la $s5, P
    la $s6, C

    li $s7, 0 #initizlize i for the loop
```

Figure 2. Encryption Initialization

The addresses of the R, K, P and C arrays are loaded to respective s registers. Afterwards the program continues with the loop to encrypt the given texts.

```
addi $sp, $sp -44 #make space for t0-2, T0-T7

lw $t1, 0($s3) #R[0]
sll $t2, $s7, 2 #Multiply the index by 4
add $t2, $t2, $s5
lw $t2, 0($t2) #P[i]

add $t1, $t1, $t2
andi $t1, $t1, 0xFFFF #Get the lower 16-bits

move $s0, $t1 #R[0] + P[i] mod 2^16
```

```
lw $t1, 0($s4) #K[0]
lw $t2, 0($s3) #R[0]

xor $t0, $t1, $t2

jal linear_func
move $s1, $t0 #L(K[0] xor R[0])
```

```
lw $t1, 4($s4) #K[1]
lw $t2, 4($s3) #R[1]

xor $t0, $t1, $t2

jal linear_func
move $s2, $t0 #L(K[1] xor R[1])

jal wfunction
sw $t0, 0($sp) #t0 stored
```

Figure 3. Input Preparation for the W Function

As the input for the W Function are retrieved from s0, s1 and s2 registers after the minor operations are completed, they are moved to their respective registers. Also, at the beginning there is space created at the stack to store the temporary values of t and T values.

```
lw $t1, 0($s3) #R[0]
add $t1, $t0, $t1 #W + R[0]
andi $t1, $t1, 0xFFFF #Get the lower 16-bits

sll $t2, $s7, 2 #Multiply the index by 4
add $t2, $t2, $s6 #Get the position of C

sw $t1, 0($t2) #Save to C
```

Figure 4. Saving of the Encrypted Text

After the operations are completed, the calculated value is saved to its respective position at the C array.

```
encryption_last_loop:

    beq $t0, 8, end_loop_encryption_last_loop

    sll $t1, $t0, 2 #index multiplied by 4
    addi $t2, $t1, 12 #position for stack pointer T values

    add $t3, $t2, $sp #position of T value
    add $t4, $t1, $s3 #position of R values

    lw $t5, 0($t3) #Load T[i]
    sw $t5, 0($t4) #Save to R[i]

    addi $t0, $t0, 1

    j encryption_last_loop
```

Figure 5. R Array Update

At the last loop, saved T values are retrieved from the stack and saved to their respective positions in the R vector and thus the R vector is updated for the next operation.

```
K[8] = {0x2301, 0x6745, 0xAB89, 0xEFCD, 0xDCFE, 0x98BA, 0x5476, 0x1032}
IV[4] = {0x3412, 0x7856, 0xBC9A, 0xF0DE}
R[8] = {0x8957, 0x112b, 0xed92, 0x2637, 0xe39e, 0x18c7, 0x96dc, 0x42fa}
P[8] = {0x1100, 0x3322, 0x5544, 0x7766, 0x9988, 0xBBAA, 0xDDCC, 0xFFEE}
```

Figure 6. Input

When the above input set is used the following output is obtained which is the correct output.

```
Done encrypting P:
0x0000926a
0x0000f9f8
0x00005bc5
0x0000b575
0x00009707
0x000006a0
0x00003407
0x000033f2
```

Figure 7. Output