Guzmán Sánchez José Emmanuel
Criptografía

MD2

1.- Con la descripción dada del algoritmo MD2 y el pseudo-código, ¿Cómo se implementaría y cuál sería el mejor lenguaje de programación para hacerlo?

Python es un lenguaje que es muy flexible con las estructuras de datos, por lo que su manipulación es sencilla. En este caso se manipulan cadenas de caracteres, lo cual puede ser convertido a lista de enteros (representación ASCII de cada carácter) y manejar esta estructura con las funciones pre-definidas de Python. Sin embargo, debido a que se hacen operaciones bit a bit, convendría usar un lenguaje de más bajo nivel, como C o VHDL, que nos permite hacer este tipo de operaciones sin mucha complicación, ya que se ahorra memoria y procesamiento que en Python se ocuparía para declarar y manipular las estructuras de datos.

Una manera sencilla de implementar este algoritmo es por módulos, es decir, distribuir las operaciones entre varias funciones y al final sólo ocupar los resultados. En este caso, conviene usar las siguientes funciones:

- ***padding***: Esta función se encargará de rellenar la cadena de entrada con bytes, para que su longitud, igual en bytes, sea múltiplo de 16.

```python
def padding(msg):
    """ Padding of the message, append n bytes with n value to the message"""

    # Number of bytes to append
    n = 16 - len(msg)%16

    # Append n bytes with valu n
    msg += [n for x in range(n)]

    return msg
```

- ***checksum***: Calcula el checksum de la cadena ya con el padding y lo agrega a ésta.

```python
def checksum(padded_msg):
    """ Append checksum (C_i) to padded message """

    checksum = [0 for i in range(16)]
    L = 0
    for i in range(len(padded_msg)//16):
        for j in range(16):
            c = padded_msg[16*i+j]
            checksum[j] = checksum[j] ^ S[c^L]
            L = checksum[j]

    padded_msg += checksum

    return padded_msg
```

- ***digest***: Se encarga de calcular el resumen de toda la cadena

```
def digest(check_msg):
    """Calculate the hash of the message"""

    X = [0 for i in range(48)]

    for i in range(len(check_msg)//16):
        for j in range(16):
            X[j+16] = check_msg[16*i+j]
            X[j+32] = X[j+16] ^ X[j]
        t=0
        for j in range(18):
            for k in range(48):
                t = X[k] ^ S[t]
                X[k] = t
            t = (t+j) % 256

    get_hex = lambda x, n: format(x, 'x').zfill(n)
    msg_dig = ''
    for i in range(16):
        msg_dig += get_hex(X[i],2)

    return msg_dig
```