Pràctica 4: La funció hash SHA-256

Funcions útils per a realitzar la pràctica:

- `int(n).to_bytes(mida_bytes,byteorder ='big')` Converteix un `Integer n` a un objecte `bytes` de mida `mida_bytes`.
- `int.from_bytes(m_bytes, "big")` Converteix un objecte `bytes m_bytes` a enter.
- `m.encode()` Converteix un objecte `string` (`m`) a un objecte `bytes`.

1. (**1 p**) Implementeu les següents funcions internes del SHA-256. Totes aquestes funcions treballen amb variables de tipus `Integer`, tant en els paràmetres d'entrada com en la sortida.
   (a) `UAB_ROTR(n,x)`: Realitza una rotació circular dels bits de `x n` posicions a la dreta.
   (b) `UAB_SHR(n,x)`: Mou els bits de `x n` bits a la dreta, omplint els nous bits amb zeros.
   (c) `UAB_Ch(e,f,g)`: Funció *choose* $(e \wedge f) \oplus (\neg e \wedge g)$
   (d) `UAB_Maj(a,b,c)`: Funció majoria $(a \wedge b) \oplus (a \wedge c) \oplus (b \wedge c)$
   (e) `UAB_Sig0(a)`: $\sum_0(a) = ROTR^2(a) \oplus ROTR^{13}(a) \oplus ROTR^{22}(a)$ on $ROTR =$ `UAB_ROTR`.
   (f) `UAB_Sig1(e)`: $\sum_1(e) = ROTR^6(e) \oplus ROTR^{11}(e) \oplus ROTR^{25}(e)$ on $ROTR =$ `UAB_ROTR`.

2. (**2,25 p**) Implementeu la funció de *padding* `UAB_SHA256_padding(m_bytes)`. Aquesta funció rebrà per paràmetre un objecte de tipus `bytes` i retornarà un altre objecte `bytes` a la sortida.
   Els càlculs per a realitzar el *padding* **els haureu de fer utilitzant operacions a nivell de bit** (p. ex: `<<`), les quals treballen sobre nombres enters. Per tant, utilitzeu les funcions mostrades més amunt (`to_bytes` i `from_bytes`) per passar de `bytes` a enters i d'enters a `bytes`.
   En aquesta funció, haureu de calcular el valor de k de la següent manera:
   $$k = 448 - (length + 1) \bmod 512$$
   on *length* és la mida en bits de l'objecte `m_bytes`. Recordeu que aquest *padding* s'ha de realitzar independentment de la mida del missatge, és a dir, encara que $length = 0 \bmod 512$.

3. (**2,25 p**) Implementeu la funció d'expansió del bloc `UAB_SHA256_extended(m_int_512)` que generi els 64 blocs de 32 bits $(W_0, W_1, \cdots W_{63})$ que utilitzareu després a la funció de compressió. Aquesta funció rebrà un **enter** com a paràmetre i retornarà la llista d'`Integers W`.

4. (**2,25 p**) Implementeu la funció de compressió `UAB_SHA256_compress(H_list,W)`, la qual transformarà els 64 blocs de 32 bits rebuts com a paràmetre (la llista `W`) en 8 blocs de 32 bits. La sortida serà també una llista d'`Integers`.

5. (**2,25 p**) Implementeu la funció de hash `UAB_SHA256(m)` que generi el hash d'una cadena de text rebuda com a paràmetre (`m`) utilitzant les tres funcions que heu programat més amunt. Aquesta funció realitzarà els passos següents:

(a) Realitzar el *padding* del missatge.

(b) Partir la cadena de bits resultant en blocs de 512 i, per a cada bloc,

(c) expandir el bloc de 512 a una llista `W` de 64 blocs de 32 bits (2048 en total), i

(d) comprimir aquesta llista en una nova llista de 8 blocs de 32 bits (256 en total).

Aquesta funció ha de retornar la llista resultant de 8 blocs de 32 bits en forma de llista d'`Integers`.

Important: Recordeu que la pràctica s'ha de programar amb SageMath. Per tant, el *kernel* del Notebook ha de ser SageMath, no Python.