Name: Drishti Bhatia **Roll Number: 07**
SYMCA **Block Chain Lab**

Practical No: 02

Aim: To implement RSA and show the encryption as well as decryption process.

Theory:

❖ Hashing

- Hashing is the process of scrambling raw information to the extent that it cannot reproduce it back to its original form.
- It takes a piece of information and passes it through a function that performs mathematical operations on the plaintext.
- This function is called the hash function, and the output is called the hash value/digest.
- Hashing works by converting a readable text into an unreadable text of secure data. Hashing is efficiently executed but extremely difficult to reverse.
- Hashing and encryption are often mistaken. Encryption is a two-way function. The plaintext can be encrypted into ciphertext and decrypted back into plaintext using a unique key.
- The difference between encryption and hashing is that encryption is reversible while hashing is irreversible.
- Cryptographic hashing provides a barrier to potential attackers. In case a malicious person tries accessing the database, the person can see the hashes. However, the attacker cannot reverse the hash value back to the actual password.
- The purpose of hashing is:
  - To verify data integrity.
  - Authentication.
  - To store sensitive data.
- A hash function is an algorithm that transforms data of arbitrary size into a fixed size output. The output is a ciphered text called a hash value or a digest. The main objective of a cryptographic hash function is verifying data authenticity.
- Types of hashing algorithms o Message digest 5 (MD5)
  - Secure hashing algorithm 1 (SHA1)
  - Secure hashing algorithm 256 (SHA256)
  - Secure hashing algorithm 512 (SHA512)

❖ SHA-256

SHA 256 is a part of the SHA 2 family of algorithms, where SHA stands for Secure Hash Algorithm. Published in 2001, it was a joint effort between the NSA and NIST to introduce a successor to the SHA 1 family, which was slowly losing strength against brute force attacks. The significance of the 256 in the name stands for the final hash digest value, i.e. irrespective of the size of plaintext/cleartext, the hash value will always be 256 bits.

The other algorithms in the SHA family are more or less similar to SHA 256. Now, look into knowing a little more about their guidelines.

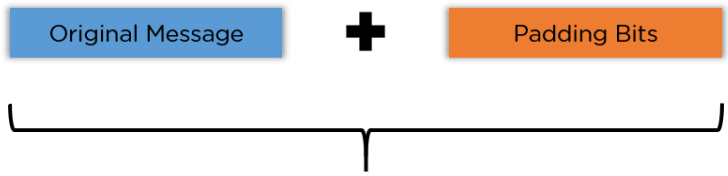
Some of the standout features of the SHA algorithm are as follows:

- Message Length: The length of the cleartext should be less than 264 bits. The size needs to be in the comparison area to keep the digest as random as possible.
- Digest Length: The length of the hash digest should be 256 bits in SHA 256 algorithm, 512 bits in SHA-512, and so on. Bigger digests usually suggest significantly more calculations at the cost of speed and space.
- Irreversible: By design, all hash functions such as the SHA 256 are irreversible. You should neither get a plaintext when you have the digest beforehand nor should the digest provide its original value when you pass it through the hash function again.

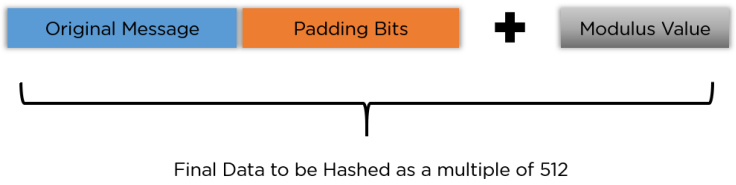
Steps:

You can divide the complete process into five different segments, as mentioned below:

- **Padding Bits**
  It adds some extra bits to the message, such that the length is exactly 64 bits short of a multiple of 512. During the addition, the first bit should be one, and the rest of it should be filled with zeroes.



- **Padding Length**
  You can add 64 bits of data now to make the final plaintext a multiple of 512. You can calculate these 64 bits of characters by applying the modulus to your original cleartext without the padding.



- **Initializing the Buffers:**
  You need to initialize the default values for eight buffers to be used in the rounds as follows:

a = 0x6a09e667

b = 0xbb67ae85

c = 0x3c6ef372

d = 0xa54ff53a

e = 0x510e527f

f = 0x9b05688c

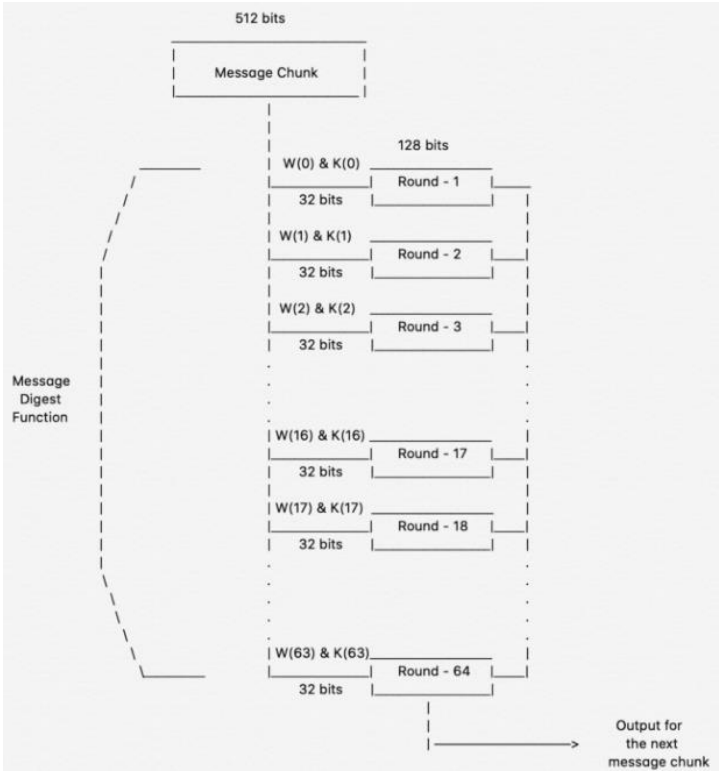
g = 0x1f83d9ab

h = 0x5be0cd19

You also need to store 64 different keys in an array, ranging from K[0] to K[63]. They are initialized as follows:

```
k[0..63] :=
   0x428a2f98, 0x71374491, 0xb5c0fbcf, 0xe9b5dba5, 0x3956c25b, 0x59f111f1, 0x923f82a4, 0xab1c5ed5,
   0xd807aa98, 0x12835b01, 0x243185be, 0x550c7dc3, 0x72be5d74, 0x80deb1fe, 0x9bdc06a7, 0xc19bf174,
   0xe49b69c1, 0xefbe4786, 0x0fc19dc6, 0x240ca1cc, 0x2de92c6f, 0x4a7484aa, 0x5cb0a9dc, 0x76f988da,
   0x983e5152, 0xa831c66d, 0xb00327c8, 0xbf597fc7, 0xc6e00bf3, 0xd5a79147, 0x06ca6351, 0x14292967,
   0x27b70a85, 0x2e1b2138, 0x4d2c6dfc, 0x53380d13, 0x650a7354, 0x766a0abb, 0x81c2c92e, 0x92722c85,
   0xa2bfe8a1, 0xa81a664b, 0xc24b8b70, 0xc76c51a3, 0xd192e819, 0xd6990624, 0xf40e3585, 0x106aa070,
   0x19a4c116, 0x1e376c08, 0x2748774c, 0x34b0bcb5, 0x391c0cb3, 0x4ed8aa4a, 0x5b9cca4f, 0x682e6ff3,
   0x748f82ee, 0x78a5636f, 0x84c87814, 0x8cc70208, 0x90befffa, 0xa4506ceb, 0xbef9a3f7, 0xc67178f2
```

- **Compression Functions**
  The entire message gets broken down into multiple blocks of 512 bits each. It puts each block through 64 rounds of operation, with the output of each block serving as the input for the following block. The entire process is as follows:



While the value of K[i] in all those rounds is pre-initialized, W[i] is another input that is calculated individually for each block, depending on the number of iterations being processed at the moment.

Python library and methods used to implement the SHA-256:

Library used: **hashlib**

Various methods:

- **SHA256** : This hash function belong to hash class SHA-2, the internal block size of it is 32 bits.
- **encode()** : Converts the string into bytes to be acceptable by hash function.
- **hexdigest()** : Returns the encoded data in hexadecimal format.

The following values are provided as constant attributes of the hash objects returned by the constructors:

- **hash.digest_size**: The size of the resulting hash in bytes.
- **hash.block_size**: The internal block size of the hash algorithm in bytes.

Code:

```
import hashlib
message="Even the darkest nights will end and the sun will rise."
result = hashlib.sha256(message.encode())
print("Message : ", message)
print("Hash Value : ", result)
print("Hexadecimal equivalent of SHA256 is: ",result.hexdigest())
print("Digest Size : ", result.digest_size)
print("Block Size : ", result.block_size)
```

Output:

```
Message :  Even the darkest nights will end and the sun will rise.
Hash Value :  <sha256 HASH object @ 0x7f37d3420510>
Hexadecimal equivalent of SHA256 is:  247abe53162406708fbdf80ab9c5652042a42ffa402bc3db8d778a6116bc9797
Digest Size :  32
Block Size :  64
```

Conclusion: We have successfully implemented SHA-256 algorithm.