# Exercises: Blockchain Cryptography

In this exercise, you shall write code to play with popular **cryptographic algorithms** using crypto libraries from your programming languages. You shall calculate hashes, calculate HMAC, derive keys from passwords, encrypt and decrypt messages.

## Calculate Hashes

Write a program to **calculate hashes** of given text message: **SHA-384**, **SHA-512**, **SHA3-512**, **Keccak-512** and **Whirlpool-512**. Write your code in programming language of choice.

**SHA-384**

|  |  |
| --- | --- |
| **Input** | **Output** |
| blockchain | 12b6459fc6b4cabb4b1990be1a78e4dc5fa79c0a0fe9aa9f0386d673cfb766171a4aaa363b8dac4c33e0ad23e4830888 |

**SHA-512**

|  |  |
| --- | --- |
| **Input** | **Output** |
| blockchain | 0bb2536b1df95e08ed016c0bae9c7ebadcafc5b1eb050de407e345dbebcc3b611f411da73b1fe5965cfec2e18698a3a91de27d047346c3820317b35f9663c9a6 |

**SHA3-512**

|  |  |
| --- | --- |
| **Input** | **Output** |
| blockchain | 75e13da2e9a446e01594ee3fda021abb1d8cfc11d8bda49735b692c5ef632285c3c937eb159e68cee47c9e53f6f721f0a4cf2099c4c01509f84de5aa38fdba79 |

**Keccak-512**

|  |  |
| --- | --- |
| **Input** | **Output** |
| blockchain | 1ab5e2e943e2fec9d5f8cc425153846591086aa4fa9b428b697606f702762fd5c074e56698432c872fb605f42dd8953824be4aadb1c1f93ea23af5f1f667bda4 |

**Whirlpool-512**

|  |  |
| --- | --- |
| **Input** | **Output** |
| blockchain | 6A14EE130EE16778CCD4F5BA9AC455DEE81D5BE3C7499BCB1C006C531BABFCFAE35C2EFA29D1BB381D99C714DA4252D87502D1325AFD64FD5D83A3DFCCE256D6 |

## Calculate HMAC

Write a program to **calculate HMAC-SHA-512** of given text **message** by given **key**. Write your code in programming language of choice.

|  |  |
| --- | --- |
| **Input** | **Output** |
| **Message:** blockchain  **Key:** devcamp | a7524b43775850de2d650ba5cce808d7fd5a1bfe2e40b11e4b6db0a92f516124d86a89fcd128491f6f7d58639ac00eb1fefc5c8317dff802371ec58d5a2bc53b |

## Derive Key by Password using Scrypt

Write a program to **calculate 256-bit key** by given string **password**, using **Scrypt**. First, generate a random 256-bit **salt**. Then derive the key from the password by using **Scrypt** (16384 iterations, block size 16, parallel factor 1). The output from your algorithm is the pair (**salt**, **derived-key**). Write your code in programming language of choice.

|  |  |
| --- | --- |
| **Input** | **Output** |
| **Password:**  p@$$w0rd~3  **Salt:**  7b07a2977a473e84fc30d463a2333bcfea6cb3400b16bec4e17fe981c925ba4f | 895cb699f600de09b203b657cd9e87a78ebdb9a972a78edde47555a7c806aeb3 |

Notes: if you use **Python** and “pip install scrypt”, you might need to install first **OpenSSL**.

## (Optional) Symmetric Encryption / Decryption (Twofish + SCrypt + HMAC)

Write a program to **encrypt** a text **message** using given **password**.

* Derive a **512-bit key** from the **password** using **SCrypt** (n=16384, r=16, p=1) with random **salt** (256 bits).
  + Split the derived key into two 256-bit sub-keys: **encryption key** and **HMAC key**.
* **Encrypt** the message using **Twofish-256** (**CBC** mode with **PKCS7** padding) using the **encryption key**.
  + Use random 256-bit **IV** (initial vector).
* Calculate message authentication code (**MAC**) using **HMAC-SHA256**(**msg**, **hmac\_key**).

**Input**: message + password.

**Output**: JSON document, holding the following assets:

* The **SCrypt** parameters: **n**, **r**, **p**, **salt** (in hex format).
* The **encrypted message** (in hex format) from the **Twofish** cipher.
* The message authentication code – **MAC** (in hex format).

|  |  |
| --- | --- |
| **Input** | **Output** |
| **Password:**  p@$$w0rd~3  **Message:**  exercise-cryptography | {"scrypt":{"dklen":64,  "salt":"7b07a2977a473e84fc30d463a2333bcfea6cb3400b16bec4e17fe981c925ba4f","n":16384,"r":16,"p":1},  "twofish":"3e7ba3686c7b5fe6f86f40b52236c8778e2921aa4356c9b4ad0f37a45702e450","iv":"433e0d8557a800a40c1d3b54f6636ff5","mac":"ffe3fdbc4216bc33296aac6221b6484e271251e33e1e25657e2bca6a6ca05caa"} |

Write a program to **decrypt** the encrypted message using given **password**.

* Derive a **512-bit key** from the **password** using **SCrypt** (n=16384, r=16, p=1) with the **salt** (from the JSON).
  + Split the derived key into two 256-bit sub-keys: **encryption key** and **HMAC key**.
* Calculate message authentication code (**MAC**) using **HMAC-SHA256**(**msg**, **hmac\_key**).
  + **Compare** the MAC with the MAC in the JSON document 🡪 correct / wrong password.
* **Decrypt** the message using **Twofish-256** (**CBC** mode with **PKCS7** padding) using the **encryption key** and the IV from the JSON.

Write your code in programming language of choice.

|  |  |
| --- | --- |
| **Input** | **Output** |
| {"scrypt":{"dklen":64,  "salt":"7b07a2977a473e84fc30d463a2333bcfea6cb3400b16bec4e17fe981c925ba4f","n":16384,"r":16,"p":1},  "twofish":"3e7ba3686c7b5fe6f86f40b52236c8778e2921aa4356c9b4ad0f37a45702e450","iv":"433e0d8557a800a40c1d3b54f6636ff5","mac":"ffe3fdbc4216bc33296aac6221b6484e271251e33e1e25657e2bca6a6ca05caa"} | exercise-cryptography |
| {"scrypt":{"dklen":64,  "salt":"abc7a2977a473e84fc30d463a2333bcfea6cb3400b16bec4e17fe981c925ba4f","n":16384,"r":16,"p":1},  "twofish":"3e7ba3686c7b5fe6f86f40b52236c8778e2921aa4356c9b4ad0f37a45702e450","iv":"433e0d8557a800a40c1d3b54f6636ff5","mac":"ffe3fdbc4216bc33296aac6221b6484e271251e33e1e25657e2bca6a6ca05caa"} | Error: incorrect ciphertext or password |