WEB 3

This Drop contract enables Reveal of NFTs at a later time:

https://github.com/thirdweb-dev/contracts/blob/main/contracts/drop/DropERC721.sol

When the creator wants to create a batch of NFTs he wants to hide under some other metadata, he needs to create a password and then he can use the “createDelayedRevealBatch” function which uses the function encryptDecrypt function of the Drop contract, please see line 61:

https://github.com/thirdweb-dev/typescript-sdk/blob/main/src/core/classes/delayed-reveal.ts

To reveal his NFTs, he needs to pass a hashproof, that is obtained thanks to the password he previously fixed, the hashproof is then checked onchain thanks again to the encryptDecrypt() function, please see line 128:

https://github.com/thirdweb-dev/typescript-sdk/blob/main/src/core/classes/delayed-reveal.ts

In both cases, the same hashing function (ecncryptDecrypt) is triggered from the drop contract:

```
function encryptDecrypt(bytes memory data, bytes calldata key) public pure returns (bytes
 memory result) {
      // Store data length on stack for later use
        uint256 length = data.length;
      // solhint-disable-next-line no-inline-assembly
    assembly {
        // Set result to free memory pointer
        result := mload(0x40)
        // Increase free memory pointer by lenght + 32
```

```
        mstore(0x40, add(add(result, length), 32))
        // Set result length
        mstore(result, length)
    }

    // Iterate over the data stepping by 32 bytes
    for (uint256 i = 0; i < length; i += 32) {
        // Generate hash of the key and offset
        bytes32 hash = keccak256(abi.encodePacked(key, i));

        bytes32 chunk;
        // solhint-disable-next-line no-inline-assembly
        assembly {
            // Read 32-bytes data chunk
            chunk := mload(add(data, add(i, 32)))
        }
        // XOR the chunk with hash
        chunk ^= hash;
        // solhint-disable-next-line no-inline-assembly
        assembly {
            // Write 32-byte encrypted chunk
            mstore(add(result, add(i, 32)), chunk)
        }
    }
}
```

Objective of the homework:

1) You need to explain how this hashing function works, line by line

2) You need to translate this function in a JS format, performing the same, no web3 calls are allowed.