CODE FLOW:

Concept:

HMAC(Hash based Message Authentication Code) is used for maintaining encryption and holding authenticity of a message. The concept is that when a sender sends a message, he uses a MAC algorithm, in our case, SHA1(hash function) to generate a MAC, a fixed bit of output message and also uses two keys to encrypt. The sender then sends the message with the MAC tag to the receiver who generates a new MAC Algorithm with a message and uses different keys to generate a MAC tag, which is compared with the sender's MAC tag to check for integrity.
The two keys used are based on outer and inner padding. The keys padd the data with XOR and generate an inner and outer keypad (ikey pad and oke ypad) which is integrated with the message to get the hash sums respectively. These keys are used with MAC algorithms to generate the MAC tag.

Credit: https://www.youtube.com/watch?v=UVPa7QNgWDo

ISHA (Implementation of Insecure Hashing Algorithm) generates the hashing functions hmac_isha computes the HMAC value for a given key and byte stream, using ISHA as the underlying hash.

1. Select a salt S and an iteration count itr
2. Select a length in octets for the derived key, dkLen.
3. Apply the key derivation function to the password, the salt, the iteration count and the key length to produce a derive key.
4. We define two fixed and different strings ipad and opad as follows (the 'i' and 'o' are mnemonics for inner and outer):
    ipad = the byte 0x36 repeated B times
    opad = the byte 0x5C repeated B times.
5. Append zeros to the end of authentication key to create a block length byte string (e.g., if authentication key is of length 20 bytes and Block Length=64, then authentication key will be appended with 44 zero bytes 0x00)
6. XOR (bitwise exclusive-OR) the block length byte string computed in previous with ipad
7. Append the stream of data 'text' to the block length byte string resulting from previous step
8. Apply H to the stream generated in step previous step
9. XOR (bitwise exclusive-OR) the block length byte string computed in First Step with opad
10. Append the H result from step (4) to the block length byte string resulting from previous step

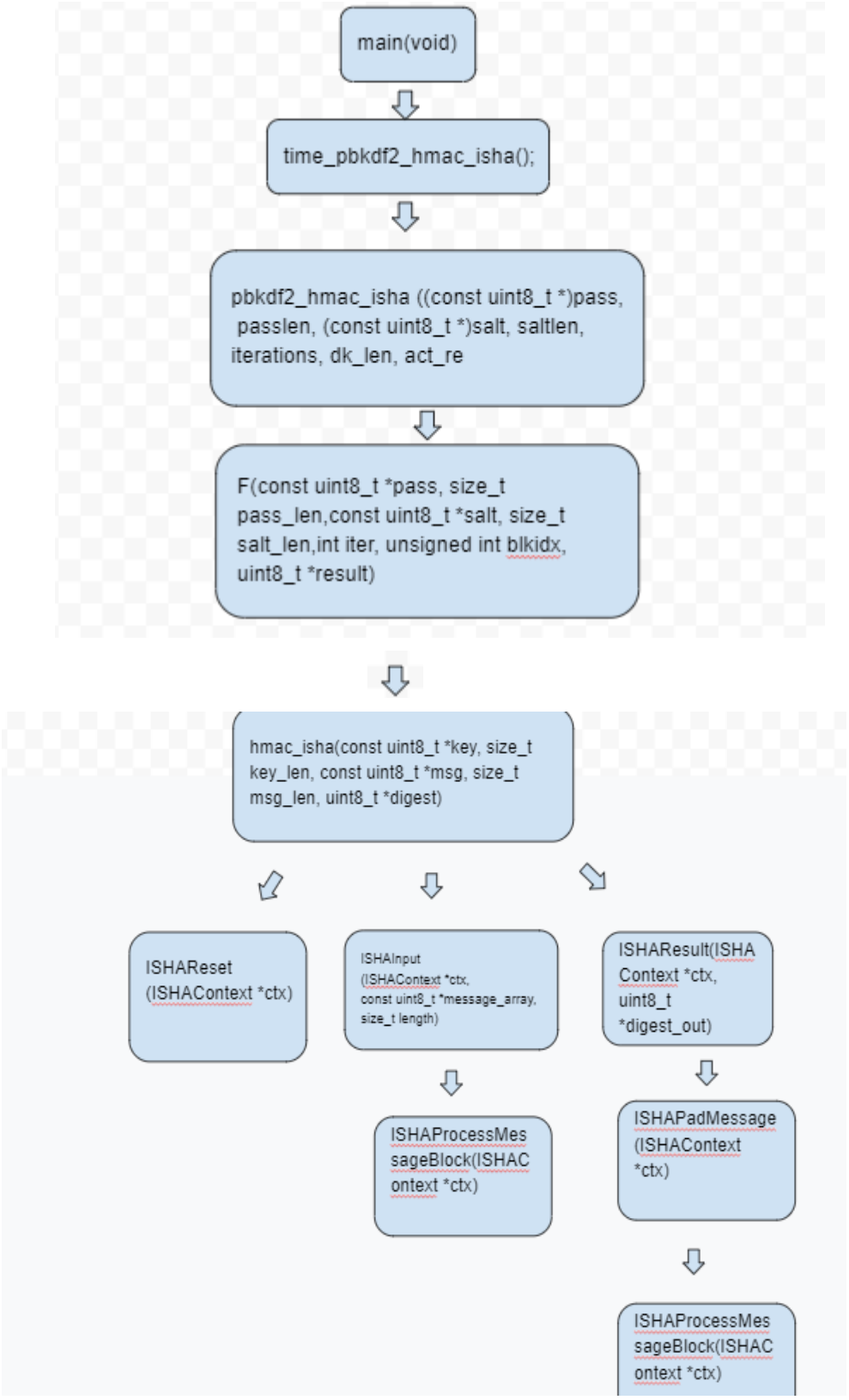
11. Apply H to the stream generated in the previous step and return the result
12. Output the derived key.

Credit: https://tools.ietf.org/html/rfc8018

CODE FUNCTIONS:

ISHAReset is used to restore an ISHA context to its default state. You can think of this as an initialization function.
ISHAInput is used to push bytes into the ISHA hashing algorithm. After a call to ISHAReset, ISHAInput may be called as many times as needed. Like SHA-1, the ISHA algorithm can hash up to 261 bytes.
ISHAResult is called once all bytes have been input into the algorithm. This function performs some padding and final computations, and then outputs the ISHA hash—a 160-bit (20 byte) value.
Hmac_isha computes the HMAC value for a given key and bytestream, using ISHA as the underlying hash. The output is also a 20-byte value.
Pbkdf2_hmac_isha can be used to derive keys of arbitrary length, based on a password and salt specified by the caller. (You can think of the password and salt as arbitrary character strings; for instance, in Wi-Fi authentication, the password is what the user considers to be the password for a given Wi-Fi network, while the salt is the SSID's name.) The output of PBKDF2 is a derived key, DK, of the length in bytes specified by the dk_len parameter.

CALL Stack Analysis:



Before optimization:
Name Size
.text. 0x00000186 hmac_isha
.text. 0x000001dc F
.text. 0x00000130 pbkdf2_hmac_isha
.text. 0x0000004c main
.text. 0x00000154 time_pbkdf2_hmac_isha
.text. 0x00000152 ISHAProcessMessageBlock
.text. 0x0000010e ISHAPadMessage
.text 0x00000060 ISHAReset
.text 0x000000c0 ISHAResult
.text. 0x000000ae ISHAInput

Post optimization:
Size analysis:
After Optimization. (release code)
.text 00000068 main
.text 0000009c time_pbkdf2_hmac_isha
.text 0000009e ISHAProcessMessageBlock
.text 00000048 ISHAInput
.text 00000320 F
.text 00000194 pbkdf2_hmac_isha
.text 00000098 hmac_isha
.text 00000034 ISHAReset
.text 000000d2 ISHAResult
.text 0000015c ISHAPadMessage

No. of Iterations:

For the code, the following iteration calculations were made (approximate results)

pbkdf2_hmac_isha() = #1
F() = #3
hmac_isha() = #4096
ISHAReset() = #24576
ISHAInput() = #49152
ISHAResult() = #24576
ISHAProcessMessageBlock() = 49152
ISHAPadMessage() = 24576