CODE FLOW:

Concept:

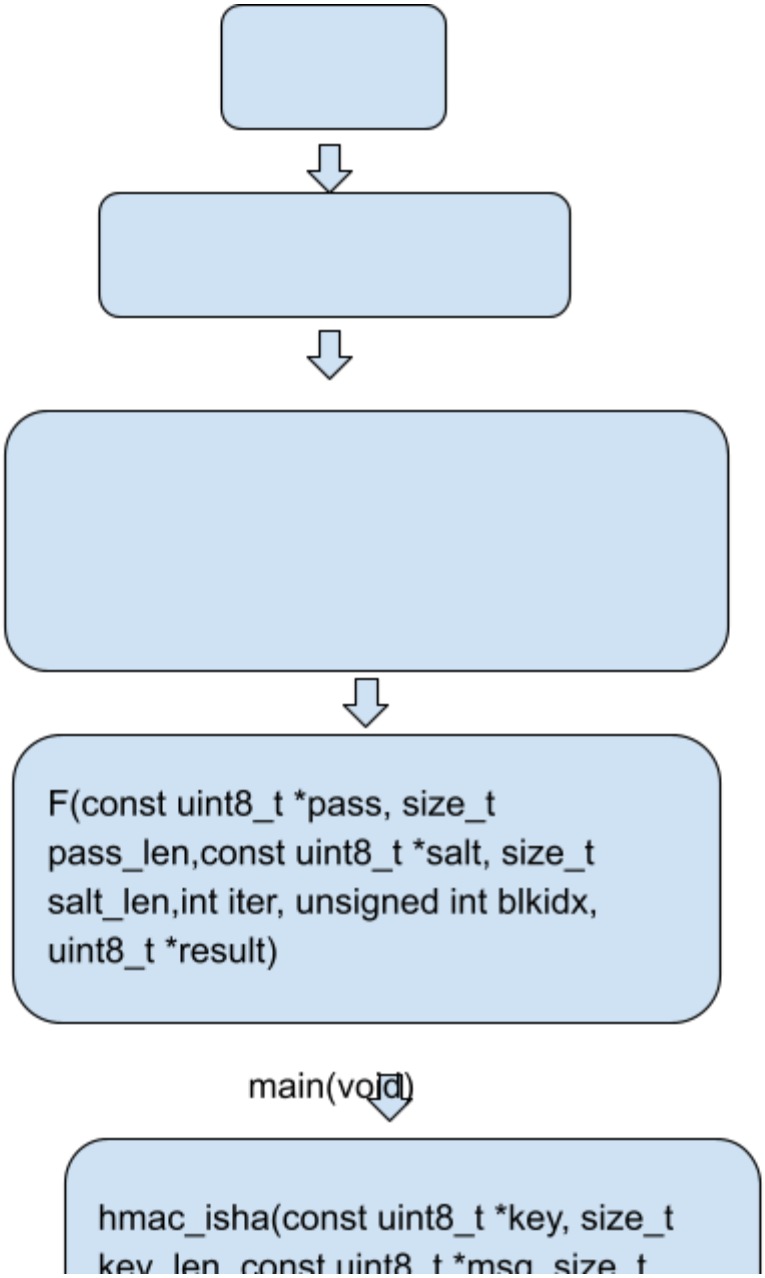
HMAC(Hash based Message Authentication Code) is used for maintaining encryption and holding authenticity of a message. The concept is that when a sender sends a message, he uses a MAC algorithm, in our case, SHA1(hash function) to generate a MAC, a fixed bit of output message and also uses two keys to 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.

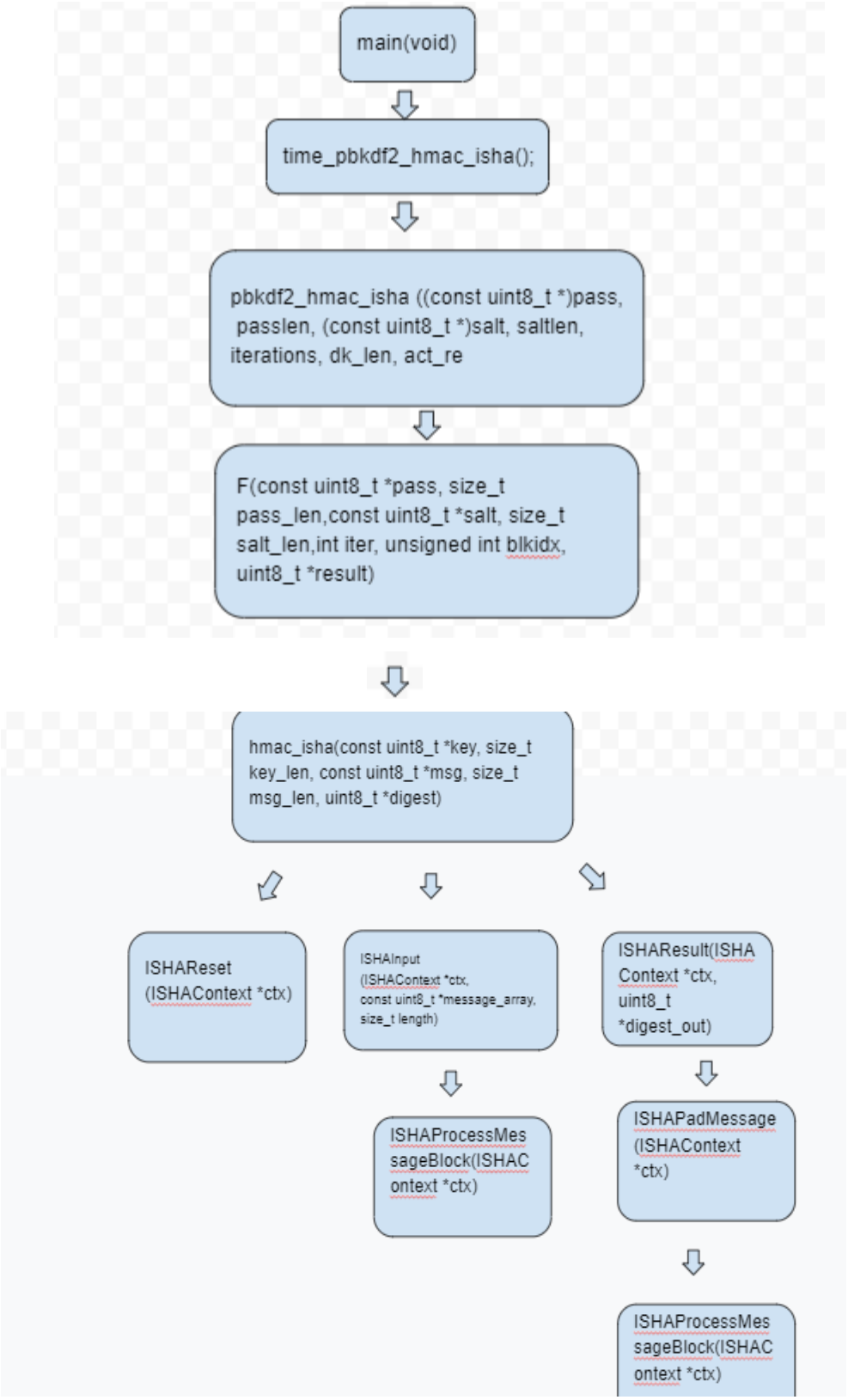
Credit: https://www.youtube.com/watch?v=UVPa7QNgWDo

CODE FLOW:

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.
`pbkdf2_hmac_isha` can be used to derive keys of arbitrary length, based on a password and salt specified by the caller.
`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:


F(const uint8_t *pass, size_t pass_len,const uint8_t *salt, size_t salt_len,int iter, unsigned int blkidx, uint8_t *result)
main(void)
hmac_isha(const uint8_t *key, size_t


main(void)
time_pbkdf2_hmac_isha();
pbkdf2_hmac_isha ((const uint8_t *)pass, passlen, (const uint8_t *)salt, saltlen, iterations, dk_len, act_re
F(const uint8_t *pass, size_t pass_len,const uint8_t *salt, size_t salt_len,int iter, unsigned int blkidx, uint8_t *result)
hmac_isha(const uint8_t *key, size_t key_len, const uint8_t *msg, size_t msg_len, uint8_t *digest)
ISHAReset (ISHAContext *ctx)
ISHAInput (ISHAContext *ctx, const uint8_t *message_array, size_t length)
ISHAResult(ISHAContext *ctx, uint8_t *digest_out)
ISHAProcessMessageBlock(ISHAContext *ctx)
ISHAPadMessage (ISHAContext *ctx)
ISHAProcessMessageBlock(ISHAContext *ctx)

Size analysis:
After Optimization.

```
.text    00000068 main
.text    0000009c time_pbkdf2_hmac_isha
.text    000000a2 ISHAProcessMessageBlock
.text    00000046 ISHAInput
.text    00000120 F
.text    000000c0 pbkdf2_hmac_isha
.text    000000a4 hmac_isha
.text    00000038 ISHAReset
.text    000000d8 ISHAResult
.text    0000015c ISHAPadMessage
```