University of Colorado Boulder

Kiran Jojare
University of Boulder/Electrical Engineering
2228 Canyon Blvd
Boulder, Colorado 80309-0000
kijo7257@colorado.edu

Technical Memo – Optimizing PKBDF2

Student Name : Kiran Jojare

Structure :

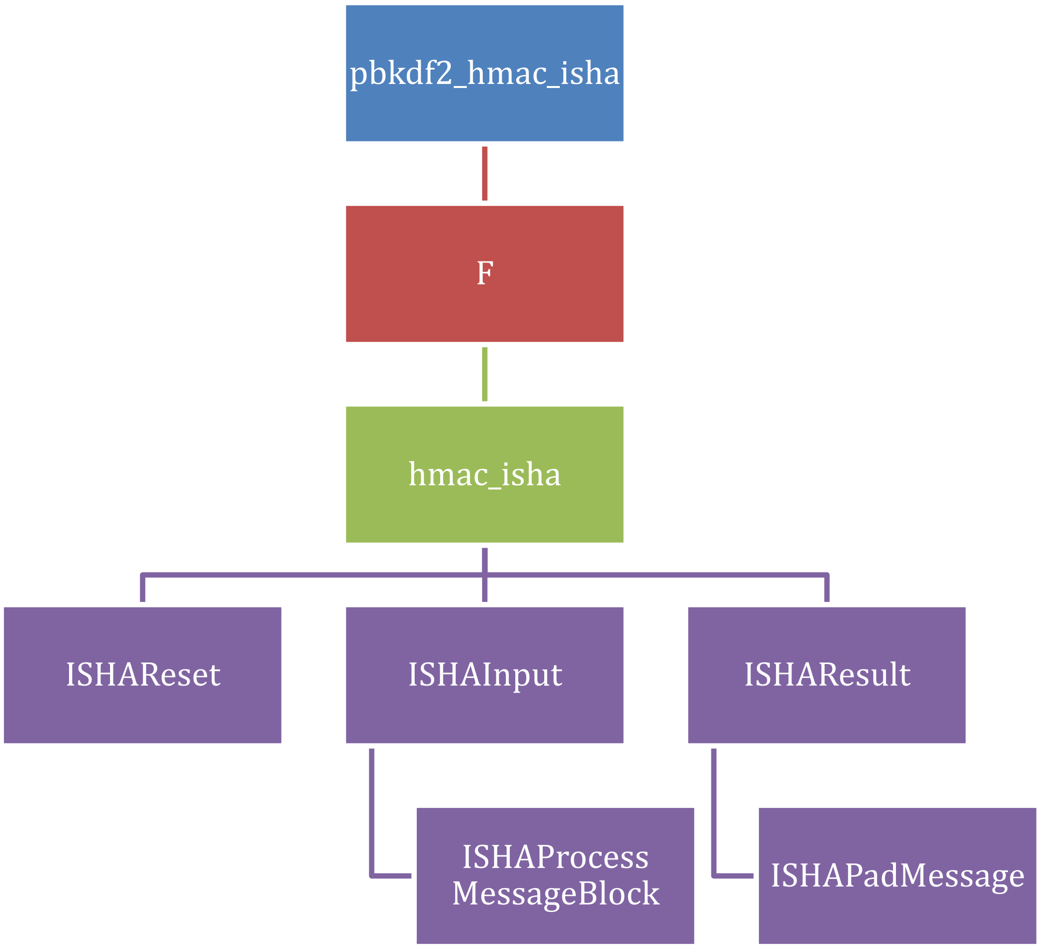


Figure 1 : Function call tree of all major functions within given code

Figure above shows the function call stack for assignment 5 code given on canvas.

The first function pbkdf2_hmac_isha() return a derived key of dkLen bytes is copied into DK. This function implements the Password-Based Key Derivation Function 2 (PBKDF2) using the ISHA hashing algorithm to derive a key from a given password and salt, with a specified number of iterations and output length. The output key is stored in the provided buffer DK.

This function then calls function F(), The F function implements the core algorithm for PBKDF2 using the ISHA hashing algorithm to derive a key from a given password and salt, with a specified number of iterations and output length. It computes the result by repeatedly applying the HMAC algorithm to the input password, salt, and block index.

F() function then calls hmac_isha(). The hmac_isha() function implements the HMAC algorithm using the ISHA hashing algorithm to provide message authentication and integrity. It generates a keyed message authentication code by performing two rounds of the ISHA hash function on the input message, using inner and outer padding keys.

hmac_isha() calls three function :

- ISHAReset
- ISHAInput
- ISHAResult

ISHAReset() this function resets the ISHAContext structure to its initial state, including setting the message length, intermediate hash values, and flags to their default values. It is typically called at the start of a new hashing operation.

ISHAResult() this function takes in a pointer to an ISHAContext struct and a pointer to an array of uint8_t (byte) data where the 20-byte digest output will be stored. The function first checks if the context is marked as "corrupted" and if so, returns without doing anything. Then it checks if the digest has already been computed, and if not, it pads the message with the appropriate padding bytes and sets the "computed" flag in the context. Finally, the function iterates over the 20-byte message digest, unpacking each 32-bit word in the digest into its four constituent bytes and storing them in the output array in big-endian byte order. The function does not return any value, but simply stores the output in the given array.

This function calls another function called ISHAPadMessage() function which is responsible for padding the message to an even 512 bits and then computing the message digest. The function first checks if the current message block is too small to hold the initial padding bits and length. If the block is too small, the function pads the block with zeros, processes it, and continues padding into a second block. Otherwise, the function pads the block with zeros. After padding, the function stores the message length as the last 8 octets.

The ISHAInput() function is used to input data to the SHA-1 algorithm. It takes in a message array and its length and processes the data in blocks of 512 bits. The function checks if the length is not zero, and if the context has not already been computed or corrupted. The data is then read byte-by-byte and stored in the message block array of the context. The length of the message is updated, and if it exceeds the maximum size, the context is marked as corrupted. When the message block array is full, ISHAProcessMessageBlock() is called to update the intermediate message digest values. The function continues this process until all the input data has been processed or the context is marked as corrupted.

The ISHAProcessMessageBlock() function is used to process the next 512 bits of the message stored in the MBlock array. It takes the ISHAContext structure as an input, which contains the message digest (MD) values and the MBlock buffer. The function starts by initializing five temporary variables (A, B, C, D, E) with the current MD values. Then it extracts 16 32-bit words from the MBlock buffer and stores them in an array called W. The function then enters a loop where it performs a series of operations on these 16 words and updates the temporary variables accordingly. This loop consists of five steps that involve circular shifts and logical operations such as AND, OR, and NOT. After completing the loop, the temporary variables are added to the current MD values and the result is stored back in the MD array. Finally, the MB_Idx variable is reset to 0 to prepare for the processing of the next 512 bits of the message.

Performance (Before Optimization)

The performance obtained before the optimization was achieved is observed as follows.

Sr No	Function	No of Iterations	Execution Time (ms)
1	pbkdf2_hmac_isha	1	8884
2	F	3	2953
3	hmac_isha	12288	8444
4	ISHAReset	24576	57
5	ISHAResult	24576	2172
6	ISHAInput	49152	4807
7	ISHAPadMessage	24576	1892
8	ISHAProcessMessageBlock	49125	2831

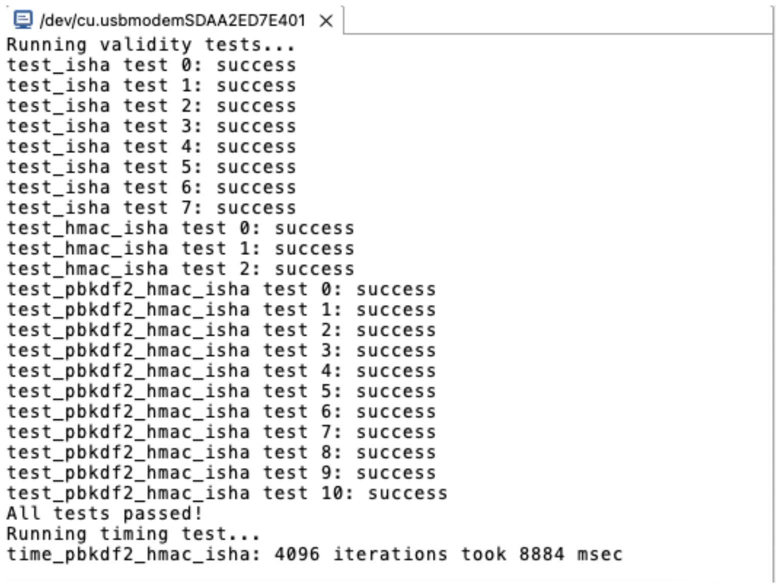
SysTick timers were used to perform the profiling. The ticktime.c functions were used to calculate the execution time of each function as well as the total execution time in the program for each function excluding time spent in test cases. This task was completed in release mode.

The timing and .text performance of the code before optimization is as follows.

Sr No	Compile Option	Speed (msec)	Size of .text (bytes)
1	-O0	8884	21036
2	-O3	2364	18292
3	-Os	3258	15084

The following method was used to calculate execution time. To begin, get timer() was called at the beginning of the function, and the time was saved in a variable. At the end of the function, get timer() was called again, and the time was saved in a different variable. The difference between each variable was saved in a third variable, which was added each time the function was called.

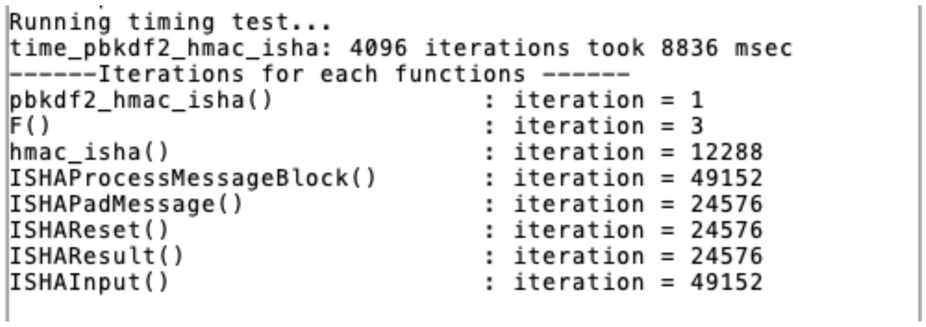
Timer 1 = timer get();
Timer 2 =get timer();
Sum   +=(Timer 2-Timer1);



```
Running validity tests...
test_isha test 0: success
test_isha test 1: success
test_isha test 2: success
test_isha test 3: success
test_isha test 4: success
test_isha test 5: success
test_isha test 6: success
test_isha test 7: success
test_hmac_isha test 0: success
test_hmac_isha test 1: success
test_hmac_isha test 2: success
test_pbkdf2_hmac_isha test 0: success
test_pbkdf2_hmac_isha test 1: success
test_pbkdf2_hmac_isha test 2: success
test_pbkdf2_hmac_isha test 3: success
test_pbkdf2_hmac_isha test 4: success
test_pbkdf2_hmac_isha test 5: success
test_pbkdf2_hmac_isha test 6: success
test_pbkdf2_hmac_isha test 7: success
test_pbkdf2_hmac_isha test 8: success
test_pbkdf2_hmac_isha test 9: success
test_pbkdf2_hmac_isha test 10: success
All tests passed!
Running timing test...
time_pbkdf2_hmac_isha: 4096 iterations took 8884 msec
```

Figure 2: Initial UART result before optimization.

The number of iterations is calculated by incrementing a global variable for each function and then same file contains the get_<function_name>_count function which returns the value of the variable indicating no of iterations of each faction before optimizations. The UART output of functions called in main to see the no of iterations of each function is as follows.

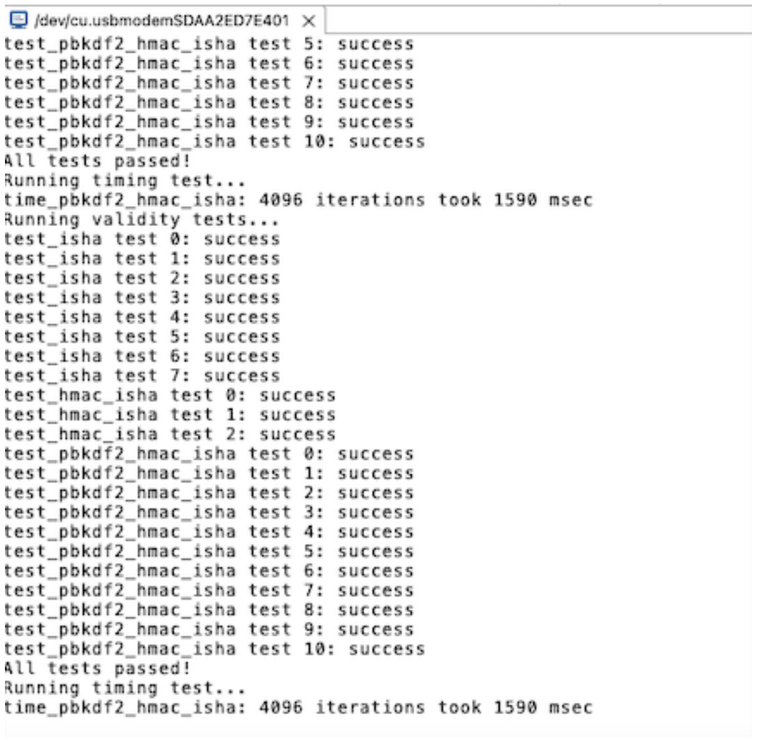


```
Running timing test...
time_pbkdf2_hmac_isha: 4096 iterations took 8836 msec
------Iterations for each functions ------
pbkdf2_hmac_isha()               : iteration = 1
F()                              : iteration = 3
hmac_isha()                      : iteration = 12288
ISHAProcessMessageBlock()        : iteration = 49152
ISHAPadMessage()                 : iteration = 24576
ISHAReset()                      : iteration = 24576
ISHAResult()                     : iteration = 24576
ISHAInput()                      : iteration = 49152
```

Figure 3 : No of iterations taken by functions defined in Structure or function call stack.

Performance After Optimization (With C)

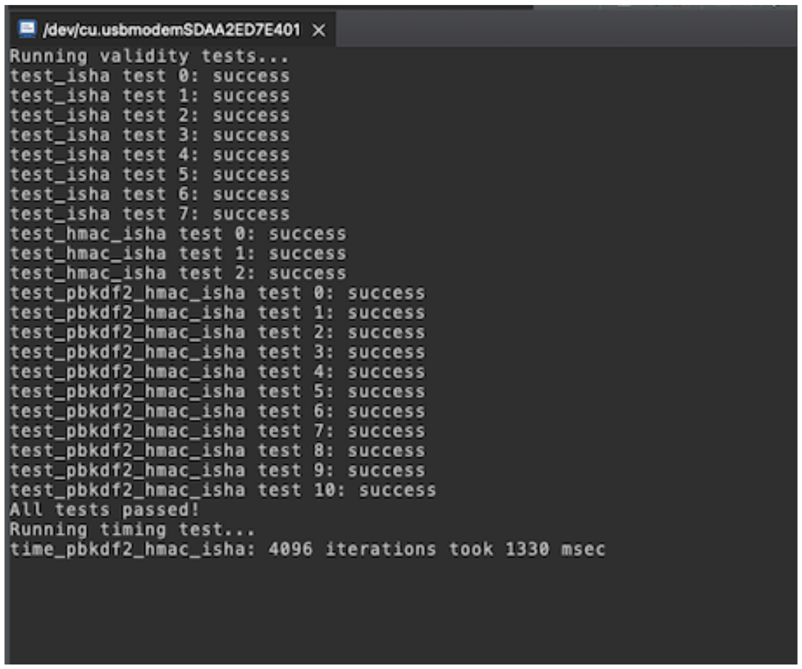
The code timing performance with optimized C version of all files is calculated as 1590 msec



```
test_pbkdf2_hmac_isha test 5: success
test_pbkdf2_hmac_isha test 6: success
test_pbkdf2_hmac_isha test 7: success
test_pbkdf2_hmac_isha test 8: success
test_pbkdf2_hmac_isha test 9: success
test_pbkdf2_hmac_isha test 10: success
All tests passed!
Running timing test...
time_pbkdf2_hmac_isha: 4096 iterations took 1590 msec
Running validity tests...
test_isha test 0: success
test_isha test 1: success
test_isha test 2: success
test_isha test 3: success
test_isha test 4: success
test_isha test 5: success
test_isha test 6: success
test_isha test 7: success
test_hmac_isha test 0: success
test_hmac_isha test 1: success
test_hmac_isha test 2: success
test_pbkdf2_hmac_isha test 0: success
test_pbkdf2_hmac_isha test 1: success
test_pbkdf2_hmac_isha test 2: success
test_pbkdf2_hmac_isha test 3: success
test_pbkdf2_hmac_isha test 4: success
test_pbkdf2_hmac_isha test 5: success
test_pbkdf2_hmac_isha test 6: success
test_pbkdf2_hmac_isha test 7: success
test_pbkdf2_hmac_isha test 8: success
test_pbkdf2_hmac_isha test 9: success
test_pbkdf2_hmac_isha test 10: success
All tests passed!
Running timing test...
time_pbkdf2_hmac_isha: 4096 iterations took 1590 msec
```

Performance After Optimization (With C and Assembly)

The code timing performance with optimized C version of all files including an assembly file for a function ISHAProcessMessageBlock is as follows.



```
Running validity tests...
test_isha test 0: success
test_isha test 1: success
test_isha test 2: success
test_isha test 3: success
test_isha test 4: success
test_isha test 5: success
test_isha test 6: success
test_isha test 7: success
test_hmac_isha test 0: success
test_hmac_isha test 1: success
test_hmac_isha test 2: success
test_pbkdf2_hmac_isha test 0: success
test_pbkdf2_hmac_isha test 1: success
test_pbkdf2_hmac_isha test 2: success
test_pbkdf2_hmac_isha test 3: success
test_pbkdf2_hmac_isha test 4: success
test_pbkdf2_hmac_isha test 5: success
test_pbkdf2_hmac_isha test 6: success
test_pbkdf2_hmac_isha test 7: success
test_pbkdf2_hmac_isha test 8: success
test_pbkdf2_hmac_isha test 9: success
test_pbkdf2_hmac_isha test 10: success
All tests passed!
Running timing test...
time_pbkdf2_hmac_isha: 4096 iterations took 1330 msec
```