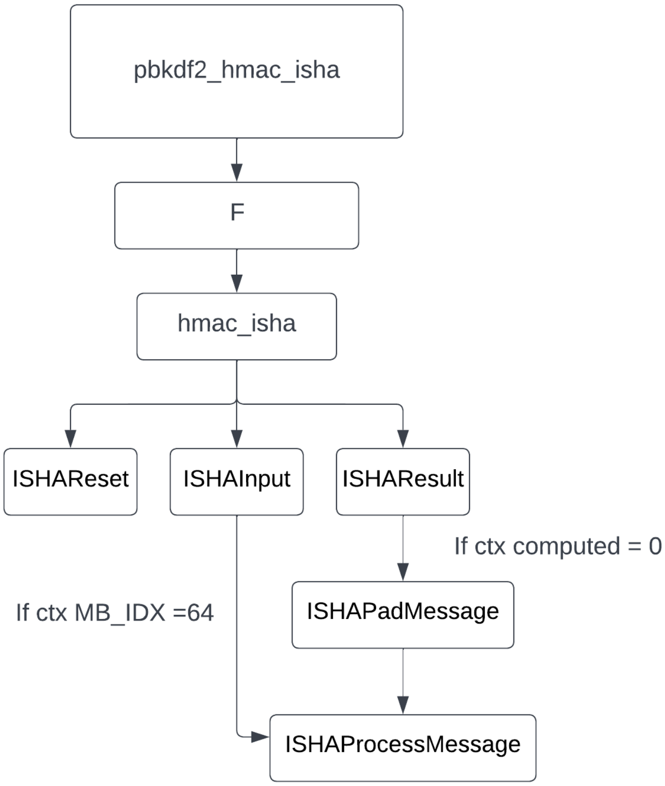
Technical Memo for PBKDF2-ISHA Optimization

The function **Call Stack** is displayed in a block diagram tree below:



To summarize the above call stack, there is a **PBKDF2-HMAC-ISHA()** function which obtains a derived key from a password and from other parameters like salt and iteration. This function calls another function **F** multiple times to compute the blocks of a derived key and concatenate the blocks to produce a derived key DK. The F function achieves this functionality by calling **hmac_isha()** once and then for iteration times. The hmac_isha takes input as secret key and message to be hashed and returns a 20-byte computed key. The hmac_isha calls **ISHAReset()**, **ISHAInput()** and **ISHAResult()** after generating keypad, ipad and opad. The ISHAReset initializes the given context back to its starting state, in preparation for computing a new message digest. The ISHAInput accepts an array of bytes as the next portion of the running ISHA hash. It calls **ISHAProcessMessage()** when the message block's array index is 64 and processes 512 bits of the message stored in the array. The ISHAResult computes the ISHA hash of the message and returns the 20-byte hash. The ISHAResult calls **ISHAPadMessage()** when context is not computed which will pad the message according to the RFC 8018 rules by filling the MBlock array accordingly. It will also call ISHAProcessMessageBlock appropriately.

Method to find the execution time for functions:

The profiling was done using SysTick timers. The functions defined in ticktime.c were used for finding out the execution time of each function and total execution time in the program for each function excluding the time spent in test cases. This activity was done in release mode.

The method followed for finding execution time was:

1) At the start of the function, get_timer() was called and time was stored in a variable
2) At the end of the function, get_timer() was called again and time was stored in another variable
3) The difference of both variable was stored in a third variable which was added each time the function was called.

```
Timer_1 = get_timer(); //At start of the function
Timer_2 =get_timer(); //At the end of the function
Sum+=(Timer_2-Timer1); //Total execution time of the funtction
```

This method was adopted for all the functions one at a time mentioned above in call stack and the total execution for each function was calculated.

Method to find the total number of times each function was called:

A variable was incremented every time the function was called and printed in main(). This method was adopted for each function mentioned in call stack.

Table consisting of functions, individual/total execution time of each function, number of times it is called:

Sr. No.	Function	Number of times called	Execution time (ms)	Total execution time (ms)
1	pbkdf2_hmac_isha	1	8754	8754
2	F	3	2912	8738
3	hmac_isha	12288	0.7	8444
4	ISHAReset	24576	0.0023	57
5	ISHAInput	49152	0.0978	4807
6	ISHAResult	24576	0.088	2172
7	ISHAPadMessage	24576	0.077	1892
8	ISHAProcessMessageBlock	49152	0.058	2831

Note: These tests were conducted before optimizing the code.

By conducting these tests, it was found that ISHAInput, ISHAResult and ISHAIpadMessage had the highest execution time and had to be targeted first to optimize.

Most of the functions were optimized and most of the changes made are mentioned as comments in the source files. The detailed changes will be mentioned in the readme.

The ISHAProcessMessageBlock() was optimized in C first before replacing with assembly code. Optimized C function can be found in readme. The assembly code was generated using disassembly feature of the IDE and then updating the assembly code to reduce the timing. The assembly code was written in a. S file and declared in a header file. The header file was included in the isha.c.