TECHNICAL MEMO – ASSIGNMENT 5

Function Call Stack

Fig: Block Diagram Indicating Functions call stack

The above flow tree shows a top-down approach of the Function call. The pbkdf2_hmac_isha calls the F function three times which in turn calls Hmac_isha to update its key digest. The hmac_isha calls the ISHAReset first, then ISHAInput and then ISHAResult. The ISHAResult Function calls ISHAPadMessage which adds required message padding upto 64 bytes and calls ISHAProcessMessageBlock to re-calculate the message according to the key digest generated by ISHAResult.

The Total Time consumed by each function execution is given below. The total execution time of the initial code without any optimization (0O) came out to be 8744 ms. The Profiling was done using SysTick timer as indicated in White's book. The hmac_isha, IshaProcessMessageBlock and ISHAInput were found to consume the most amount of execution time in the whole software application of SHA-1 Algorithm implementation

The Profiling was done by using the Get_timer function in each function's first line of execution and last line. Then, the difference was added to the count and using the time each interrupt is triggered, the total execution time of each function were found. Following is the table indicating the number of each Function calls and the time consumed by them.

Function	Total time in msec by each function	Number of Times function called
F	8738	3
pbkdf2_hmac_isha	8738	1
hmac_isha	8438	12288
isha_result	2166	24576
isha_input	4806	49152
ishaPadMessage	1942	24576
ishaProcessMessageBlock	2833	49152

As Hmac_Isha function consumed the most time and is also called a lot, it was clear that it needs to be optimized to even reduce the overall time to even 25% of the given time. So, the focus was immediately on hmac_isha which calls functions IshaReset and IshaInput multiple times, which in turn calls ISHAprocessMessageBlock multiple times. So, it was also clear that if we could reduce the functions count to ISHAinput the calls to other function would reduce as well making a drastic change in the total execution time.

The ISHAResult could also be optimized by using the three easiest ways to optimizing

1) Removing Loop Invariant code
2) Loop unrolling
3) Using built-in functions that are highly optimized over the years after multiple updates.

The biggest Optimization could be done in hmac_isha due to high percentage of Loop Invariant code. The built-in functions could radically change the execution time of the instructions as they run on the same concept. As the task was to reduce the time by a huge margin, Loop Unrolling were limited as it doesn't affect the overall results by a lot.