Implementing Cipher APIs in Inter IXP 2400

Sang-Su Lee*, Min-Ho Han*, and Jeong-Nyeo Kim*

* Network Security Department, ETRI, Korea
(Tel : +82-42-860-1613; E-mail: {sangsu, mhhan, jnkim}@etri.re.kr)

Abstract: In this paper, we presented our implementation of 3DES and HMAC-MD5 processing functionality in Intelâ IXP 2400 platform. It can be used as encryption and authentication engine for VPNs such as IPsec and SSL.

Keywords: Network Processor[1], DES[2], 3DES[2], MD5[3], HMAC[4], IPsec[5]

1. INTRODUCTION

For recent broadband network environment, network devices, especially network security appliances such as firewall, IDS(Intrusion Detection System) or IPS(Intrusion Prevention System), and VPN gateway, need H/W based design. NP(Network Processors) are very common solution to reduce the processing load from the host processor and accelerate the performance associated with networking.

In this paper, we presented the design issues to implement the cipher module which processes cryptographic algorithms, 3DES and HMAC-MD5, in Intelâ IXP 2400 platform. The module can be used for IPsec or SSL.Each paper must be divided into two parts. The first part includes the title, authors' name, abstract and keywords. The second part is the main body of the paper.

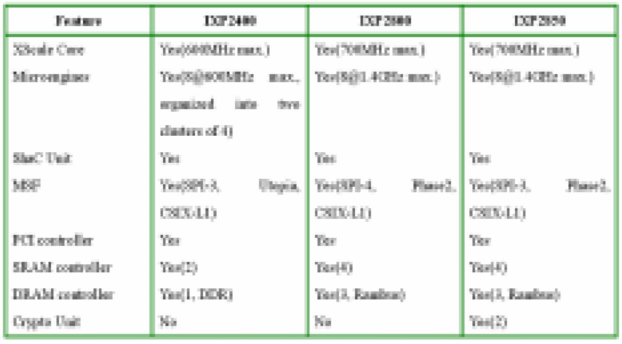
2. TARGET PLATFORM

IXP 2400 is one of network processors released by Intelâ and consists of 9 programmable processors: one Intel XScale core and 8 second-generation Microengines all on the same die. The Intelâ XScale core is an advanced Reduced Instruction Set Computer (RISC) machine that is compliant with ARM Architecture V5STE, general-purpose processor. The micro-engines are RISC processors optimized for fast-path packet processing. And, IXP 2400 supports its own programming languages: Microengine Assembly and Microengine C.

Table 1 shows the major components of the IXP 2400 and the newly released IXP 2800 and IXP 2850.

There are DRAM, SRAM controllers and Scratchpad memories inside of IXP 2400. However, each Microengine has its own local memory, too. Among them, local memory has the shortest access time, and we used it as main memory for our implementation. Figure 1 shows the functional units of the IXP 2400.

Table 1 The major components of the IXP 2400/2800/2850.



Feature	IXP 2400	IXP 2800	IXP 2850
XScale Core	Yes(600MHz max.)	Yes(700MHz max.)	Yes(700MHz max.)
Microengines	Yes(8@600MHz max., organized into two clusters of 4)	Yes(8@1.4GHz max.)	Yes(8@1.4GHz max.)
ShaC Unit	Yes	Yes	Yes
MSF	Yes(SPI-3, Utopia, CSIX-L1)	Yes(SPI-4, Phase2, CSIX-L1)	Yes(SPI-3, Phase2, CSIX-L1)
PCI controller	Yes	Yes	Yes
SRAM controller	Yes(2)	Yes(4)	Yes(4)
DRAM controller	Yes(1, DDR)	Yes(3, Rambus)	Yes(3, Rambus)
Crypto Unit	No	No	Yes(2)

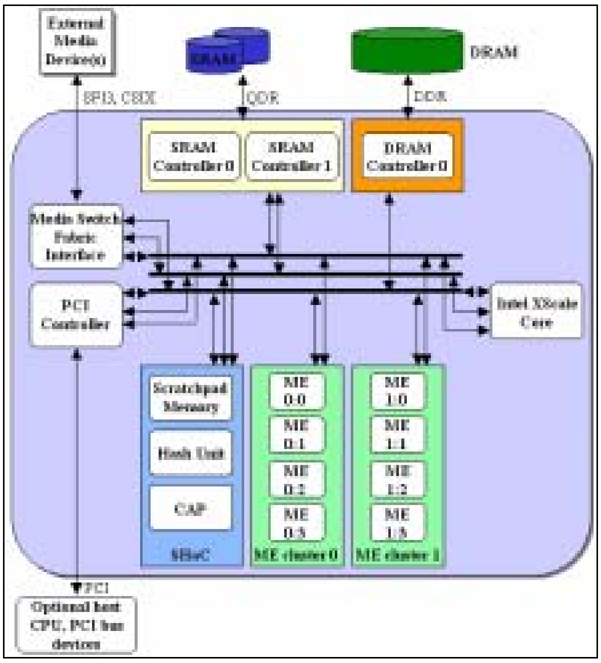


Fig. 1 The caption should be placed after the figure.

We divided the local memory into several segments to store data and look-up tables. For example, Sbox for DES and T-values for MD5, a key for 3DES and a key for HMAC-MD5 are stored in proper segments.

3. Cipher API implementation in IXP 2400

3.1 Local memory Usage

The local memory inside each of Microengines has 4 bytes width and 2560 bytes length and we divided it several segments to store date related with each cipher algorithm. The figure 2 shows the detailed usage of local memory in our implementation.

3.1.1 LM_MD5_IPAD_BASE

This is the start address from which 64 bytes date obtained by XOR with hash key and the constant value of 0x36363636 for HMAC-MD5 is stored.

3.1.2 LM_MD5_DATA_BASE

This is the start address from which data is stored and its maximum is 1536 bytes. Notice that date includes the original information to be hashed and padding bits MD5 algorithm requires.

3.1.3 LM_MD5_OPAD_BASE

This is the start address from which 64 bytes date obtained by XOR with hash key and the constant value of 0x5c5c5c5c for HMAC-MD5 is stored.

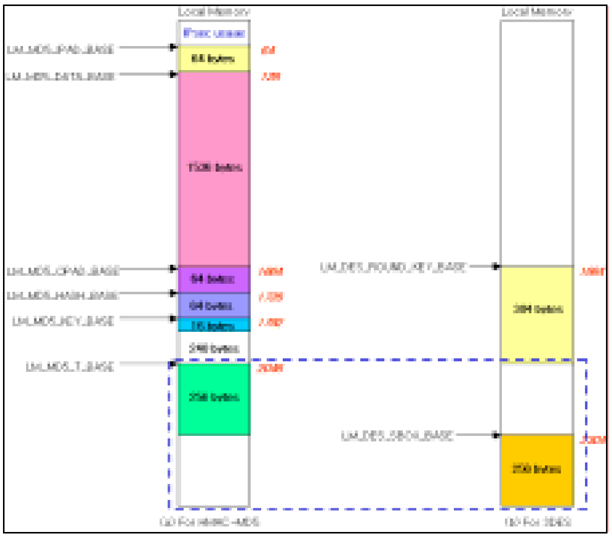


Fig. 2 The local memory usage map.

3.1.4 LM_MD5_HASH_BASE

The result of single MD5 is stored from this address. Actually, the first hash result using inner pad, key, and data is stored in it. And the next hash result using outer pad, key, and the first hash value overrides this segment. For IPsec, upper 12 bytes out of the final hash value are used as HMAC-MD5 result.

3.1.5 LM_MD5_KEY_BASE

From this address, 128 bits key of HMAC-MD5 algorithm is stored.

3.1.6 LM_MD5_T_BASE

The T-values defined in MD5 algorithm are stored from this address. For this, 256 bytes out of the local memory space are required.

3.1.7 LM_DES_SBOX_BASE

Sbox values defined in DES algorithm are stored from this address. For Sbox, 256 bytes of local memory are used.

3.1.8 LM_DES_ROUND_KEY_BASE

The round keys for 3DES are stored from it. This segment is shared with segment start from LM_MD5_OPAD_BASE. Thus, in our implement HMAC-MD5 and 3DES are not processed at the same time.

3.2 Implemented cipher APIs

Implemented functionalities are coded as macro function style using IXP 2400-native macro-assembler. However, any applications implemented using the assembler can reuse this APIs on their purposes.

3.2.1 *insert_T* macro function

This macro stores T-values in local memory area starting from LM_MD5_T_BASE. Thus, it doesn't need to be called in every application using our implementation for MD5 or HMAC-MD5.

3.2.2 *get_HMAC_MD5* macro function

This is the main macro to obtain HMAC-MD5 result. *get_MD5* is another macro which processes pure MD5 algorithm and called by *get_HMAC_MD5* macro inside. XORed values using 128-bit hash key and inner pad, and data to be hashed are processed by *get_MD5* and its result is stored from LM_MD5_HASH_BASE. However, XORed values with the hash key and outer pad are stored from LM_MD5_OPAD_BASE.

Finally, *get_MD5* performs hashing using the XORed value and previous hashing value. The first 96-bit value of the second hashing is used for authentication in IPsec. The figure 3 shows the processing step of it.

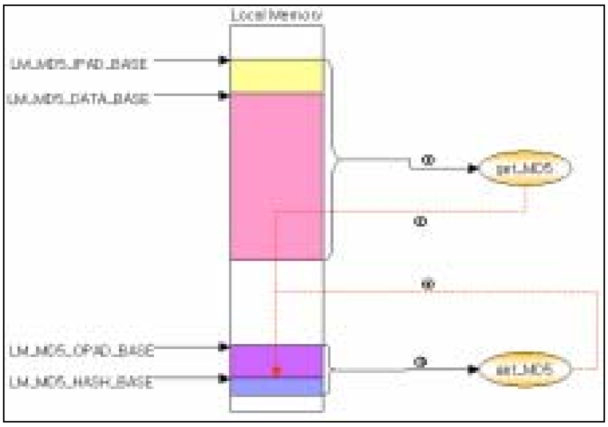


Fig. 3 Processing steps of *get_HMAC_MD5* macro.

3.2.3 *get_MD5* macro function

This macro performs pure MD5 hashing algorithm and returns 128 bits hash value. In our implementation, data to be hashed is re-arranged to big-endian order because Microengines treat the date in big-endian manner.

3.2.4 *initiate_SBOX* macro function

This macro places the Sbox values from LM_SBOX_BASE of local memory. In general, DES has 8 Sboxs and each of them needs at least 32 bytes. Thus, 256-byte memory area is needed.

3.2.5 *tri_key_schedule* macro function

This macro derives round keys of 3DES from 192 bits original key. In IPsec, this key value would be agreed through IKE of two peers.

This macro uses another internal macro named *key_schedule* which generates round key from 64 bits key value for DES. In detail, Let's define the 192 bits key as KEY and three of 64 bits segment as key1, key2, and ket3. Then KEY can be considered as the concatenation of the three segments. *key_schedule* macro derives round key of DES from each of the segment, so it should be called 3 times in *tri_key_schedule* as showed in figure 4.

Total size of round keys is 384 bytes. Figure 4 shows round key generation steps.

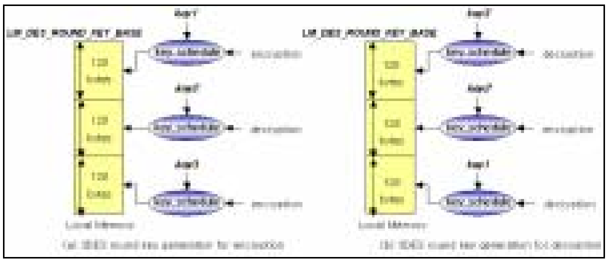


Fig. 4 Processing steps of *tri_key_schedule* macro.

3.2.6 *tri_DES_CBC* macro function

This macro uses another internal macro, *Cipher*, which implements pure DES encryption and decryption and returns 64 bits result. *Cipher* used round keys derived by tri_key_schedule macro and called three times by *tri_DES_CBC* which supports 3DES in EDE. Figure 5 shows the processing steps.

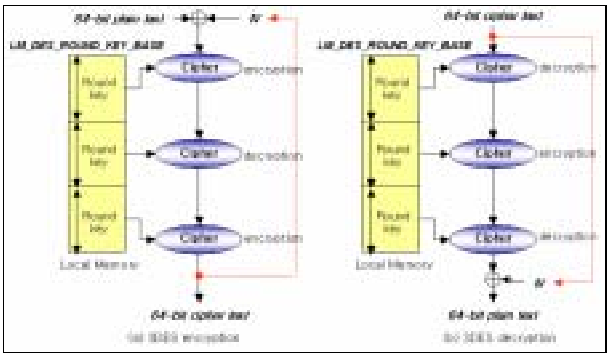
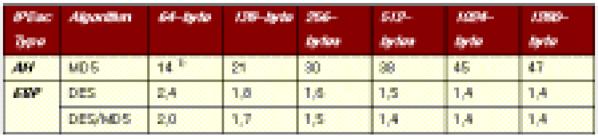


Fig. 5 Processing steps of *tri_DES_CBC* macro

4. TEST PERFORMANCE

With the cipher module explained in previous section, we implemented IPsec VPN engine in IXP 2400, and its performance obtained by test is summarized in table 2. Notice that the unit of the value is Mbps.

Table 2 Test performance of IPsec module using our APIs.



IPSec Type	Algorithm	64-byte	128-byte	256-bytes	512-bytes	1024-byte	1280-byte
AH	MD5	14	21	30	38	45	47
ESP	DES	2,4	1,8	1,6	1,5	1,4	1,4
	DES/MD5	2,0	1,7	1,5	1,4	1,4	1,4

Zero-loss Throughput across an IPSec Tunnel:
Ixia 1Gbit/s Fast Ethernet (Uni-Directional UDP Packets)

5. CONCLUSTIONS

In this paper, we described our implementation of 3DES and HMAC-MD5 functionalities in Intel IXP 2400 platform. We used local memory of each Microengine of IXP 2400 and implemented macro function style APIs to process the algorithms as.

REFERENCES

[1] Bill Carlson, *Intel Internet Exchange Architecture and Applications – A Practical Guide to IXP2XXX Network Processors*, Intel Express, 2003.

[2] Shneier. B., *Applied Cryptography*, John Wiley, NY, 1994.

[3] R. Rivest, “The MD5 Message-Digest Algorithm,” *IETF RFC 1321*, 1992.

[4] H. Krawczyk, M. Bellare, and R. Canetti, “HMAC: Keyed-Hashing for Message Authentication,” *IETF RFC 2104*, 1997.

[5] S. Kent and R. Atkinson, “Security Architecture for the Internet Protocol,” *IETF RFC 2401*, 1998.