情 報 保 護 學 會 誌
第 16 卷 第 3 號, 2006. 6

IPsec 구현 방법 및 SoC 소개

김 종 혁* 정 영 조** 조 인 현*** 김 현 철****

요 약

IP 네트워크의 보안으로 가장 널리 사용되고 있는 표준은 IPsec 방식이다. 일반적으로 IPsec의 구현은 통신 장비 내에 소프트웨어 방법을 사용하거나, 하드웨어 암호가속기를 사용하여 구현한다. 소프트웨어 방식의 구현은 저속의 통신에서 주로 이용되며, 고속의 경우 하드웨어 암호가속기를 사용하고 있다. 하드웨어 암호가속기를 사용하는 경우에도 시스템의 구조에 따라 암호가속기의 성능을 충분히 발휘하지 못하는 경우가 대부분이다. 본 논문에서는 CPU와 IPsec 엔진을 하나로 통합해 최적의 성능 (Wire-speed)을 발휘하도록 구현한 SoC인 FSC2003을 소개한다.

Ⅰ. 서 론

최근 인터넷이라고 불리는 IP 네트워크의 사용이 범용적인 용도에서 정부, 기업, 금융, 등을 행하는 통합 네트워크로 확대되고 있다. 많은 사람들이 잘 아는 것처럼 IP 네트워크는 쉽게 노출되어 있어 비밀을 요하는 일을 수행하는 데 있어 취약한 면을 가지고 있다. 취약한 IP 네트워크를 보안하는 표준 방법으로 IPsec이라는 보안 프로토콜이 있는데, 이 방식을 현재 가장 많이 사용하고 있다. 국내의 경우 금융권에서 여러 지점을 연결하는 데 비싼 전용망 대신 저렴한 브로드밴드 네트워크를 활용해 금융 데이터를 처리하고 있으며 네트워크 보안을 IPsec을 통해 해결하고 있다.

IPsec을 구현하는 방법으로 크게 소프트웨어 방식, 암호 또는 IPsec 전용 암호가속기 방식, CPU와 IPsec 암호가속기 및 네트워크 기능이 통합된 SoC방식이 있다. IPsec을 사용하는 목적에 따라 위의 방식 중 적절한 방식을 선택하여 구현하게 된다.

저속의 IPsec을 구현하는 경우 대부분은 소프트웨어 방식을 택하고 있으며, 고속의 IPsec을 구현하는 경우 암호 또는 IPsec 전용 암호가속기를 사용한다. 특히 IPsec 네트워크 장비를 구현하는 경우 들어오는 IP 패킷을 IPsec 패킷으로 변환하여 전달하는 역할(또는 반대로)을 해야 하는데 이 경우 IPsec처리 문제와 더불어 패킷을 전달하는 성능 또한 요구된다.

SoC란 특정한 목적을 가지고 여러 칩 또는 기능을 하나로 구현하는 형태를 말한다. SoC는 단순히 칩의 크기와 소비전력을 줄이기 위한 목적으로 여러 컴포넌트를 하나로 묶어 구현하는 형태와 성능을 최대로 높이기 위해 여러 컴포넌트를 유기적으로 동작하도록 소프트웨어적인 요소를 하드웨어 로직으로 구현하여 같이 묶어 주는 형태의 구현으로 나눌 수 있다.

Ⅱ장에서는 여러 구현 방법을 , Ⅲ장에서는 SoC 형태의 구현을, Ⅳ장에서는 SoC의 성능을 Ⅴ장에서는 결론을 내리도록 한다.

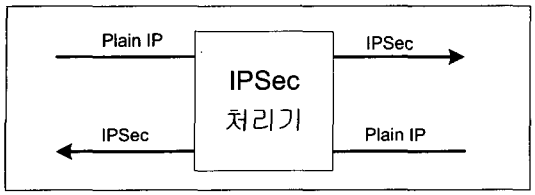
Ⅱ. IPsec의 구현 형태

IPsec 통신 장비의 성능에 영향을 미치는 요소는 패킷을 전달하는 능력과 일반 IP 패킷을 IPsec 패킷으로 변환 (또는 반대)하는 능력이라 할 수 있다.

본 장에서는 IPsec 구현 방법인 소프트웨어 방식과 암호가속기를 활용한 방식을 소개한다.

IPsec 구현에서 성능에 미치는 요소는 일반 패킷의 전송 성능과 IPsec 처리 성능이 있다. IPsec 성능은 현재 가장 널리 보급되어 사용중인 FE(Fast

* (주) 퓨쳐시스템 책임연구원(jhkim@future.co.kr)
** (주) 퓨쳐시스템 홈네트워크사업본부 본부장(ycchung@future.co.kr)
*** (주) 퓨쳐시스템 선임연구원(ihcho@futuer.co.kr)
**** (주) 퓨쳐시스템 선임연구원(hckim@future.co.kr)



〔그림 1〕 IPsec 처리기의 동작

Ethenet, 100Mbps 회선 속도) 상에서 측정하는 것으로 한다.

IPsec의 성능 측정은 가장 연산 능력이 많이 필요한 ICV(Integrity Check Value)가 포함된 ESP(Encapsulating Security Payload) 변환을 대상으로 한다. IPsec 방식은 ESP와 AH (Authentication Header) 방식이 있으며, ESP의 경우 추가적으로 ICV 값을 첨부할 수 있다.〔3〕〔4〕

ICV가 포함된 ESP 변환을 적용하는 경우 일반 IP 패킷보다 약 50바이트가 증가되므로 일반 패킷을 기준으로 100Mbps의 성능을 제공할 수 없다. 원본 IP 패킷에 IPsec을 적용하면 패킷의 크기가 어떻게 변하는지 조사하고, 이 과정에서 이론적인 IPsec의 최대 성능이 얼마가 나오는지를 알아보자.

다음은 일반 패킷을 IPsec 패킷으로 변환하는 경우 크기의 변화를 계산하는 수식이다.〔1〕〔2〕〔3〕〔4〕

$$\begin{aligned} &IPFrame = MH + IH + IP + CRC \\ &where\ MH = MACHeader(14bytes) \\ &\qquad IH = IPHeader(20bytes) \\ &\qquad IP = IPPayload \\ &\qquad CRC = Ethernet\ CRC(4bytes) \end{aligned} \tag{1}$$

$$\begin{aligned} &IPSecFrame = MH + IHn + EH + IV \\ &\qquad + Encrypt(IH + IP) + A + CRC \\ &where\ IHn = new\ IPheader(20bytes) \\ &\qquad EH = ESPHeader(8bytes) \\ &\qquad IV = initial\ vector\ \text{for}\ ESP \\ &\qquad A = ICV(variable) \end{aligned} \tag{2}$$

수식 (1)은 IP 패킷에 대한 표현식이고 (2)의 수식은 IP 패킷이 IPsec 패킷으로 변환되는 경우 패킷의 내용이 어떻게 달라지는지를 나타낸다.

(2)의 수식에서 *IV*와 *Encrypt()*의 데이터 길이는 암호 알고리즘에 따라 다르게 정의된다. DES, 3DES는 8바이트 *IV*를 사용하고, AES, SEED의 경우 16바이트 *IV*를 사용한다. *Encrypt()*는 암호 알고리즘의 블록크기의 배수로 결정된다. 암호 알고리즘에 따른 블록 크기는 DES와 3DES는 8바이트이고, AES와 SEED는 16바이트이다. 따라서 DES나 3DES를 사용하는 경우 *Encrypt()*는 8의 배수로 결정된다. *Encrypt()*함수 적용시 ESP 패딩에 따른 길이의 증가도 포함된다. *A*는 ICV를 의미하며 가변 길이를 갖는다. 보통의 경우 96비트(12바이트)를 사용한다.〔4〕〔5〕〔6〕〔7〕

위의 수식에 따라서 계산한 IP frame과 IPsec frame의 패킷 길이의 상관관계를 정리하면 〔표 1〕과 같다.

〔표 1〕 IP 패킷 크기와 IPsec 패킷 크기 변환표. 이론적인 최대 IPsec 성능을 나타낸다.(A=12바이트)

IP 패킷크기 (바이트)	암호알고리즘 블록 (8바이트)		암호 알고리즘 블록 (16바이트)	
	IPsec 패킷크기 (바이트)	이론적인 최대 전송률 (%)	IPsec 패킷크기 (바이트)	이론적인 최대 전송률 (%)
64	114	62.69	122	59.15
128	178	74.75	186	71.84
256	306	84.66	314	82.63
512	562	91.41	570	90.17
1024	1074	95.43	1082	94.74
1400	1450	95.60	1450	96.60

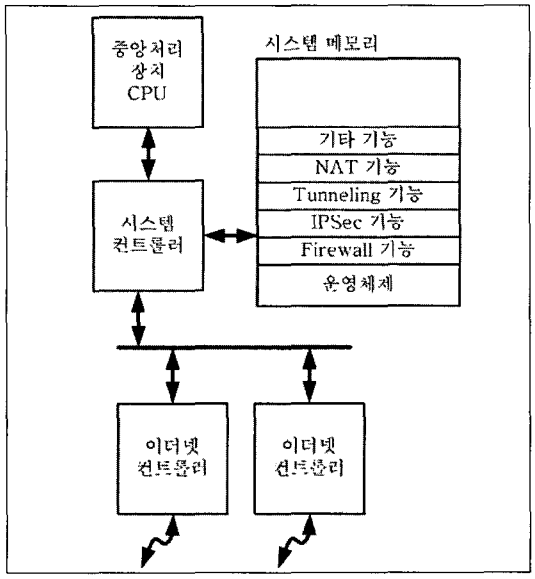
〔표 1〕에서 IP 패킷이 64바이트인 경우 8바이트 블록 암호 알고리즘을 적용하면 IPsec 패킷은 114바이트가 되며, 이때 100Mbps 패스트 이더넷을 통해 전송할 수 있는 최대 IP 패킷은 62.69Mbps이다. 이 이상의 IP 패킷은 모두 유실된다. 즉 62.69Mbps의 IP 패킷을 전송하면 IPsec을 적용한 후 IPsec 패킷은 100Mbps의 전송율을 가지게 된다.〔2〕〔8〕〔9〕

1. 소프트웨어 방식의 구현

소프트웨어 방식이란 시스템 내에 존재하는 CPU에서 실행되는 프로그램을 작성하여 구현하는 방식으로 일반적으로 CPU, 시스템 버스를 제어하기 위한 시스템 컨트롤러, 메모리, FE 컨트롤러로 구성된다.

소프트웨어 방식은 IP 패킷의 처리와 IPsec 적용 혹은 반대의 기능을 모두 CPU에서 담당하는 것으로 〔그림 2〕의 하드웨어 구조에서 동작한다.

소프트웨어 방식의 구현은 전적으로 CPU와 시스템 컨트롤러에 의해 패킷 처리 성능이 결정된다.



〔그림 2〕 소프트웨어 방식 IPsec 처리 하드웨어 구조

이 방식에서 패킷의 이동 경로는 네트워크->FE컨트롤러->메모리->CPU(패킷 읽기, IPsec처리, 패킷 쓰기)->메모리->FE컨트롤러->네트워크이다.

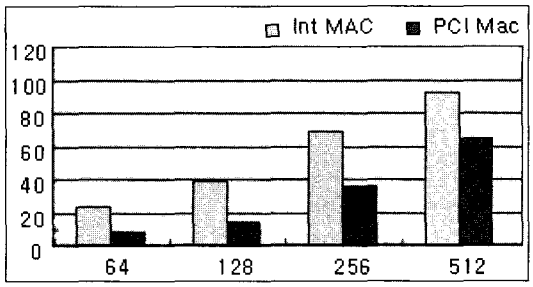
IP 패킷은 FE컨트롤러가 수신하여 메모리로 적재하며, CPU에서 수신된 IP 패킷을 각 네트워크 기능을 적용하고 최종적으로 IPsec을 적용한다. 이때 패킷 처리 성능은 수신된 IP 패킷을 FE 컨트롤러로부터 시스템 메모리로 전송되는 과정(IP 패킷을 전송하는 경우 반대의 과정)과 시스템 메모리에 적재된 패킷을 CPU에서 처리하는 과정에 의해 결정된다.

일반적으로 소프트웨어 방식의 패킷 처리 성능은 전체 통신량과의 관계보다는 전송된 패킷의 개수에 의존적이다. 즉 모든 패킷은 CPU를 거쳐서 처리되며, CPU는 각 패킷 당 동일한 기능을 수행하게 되므로 CPU 사용량은 처리한 패킷의 개수와 밀접한 관련이 있다.〔11〕

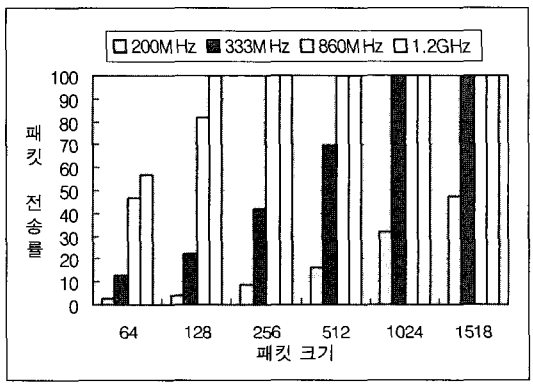
또한 FE컨트롤러가 PCI 등의 인터페이스 방식에 의존하지 않고 시스템 메모리를 직접 접근하는 경우 더 좋은 성능을 보인다.〔13〕

〔그림 3〕은 PCI 방식의 FE컨트롤러(PCI MAC)와 내장 FE컨트롤러(Int MAC)의 성능 비교 자료이다.

소프트웨어 방식에서 IPsec을 적용하지 않고 (CPU에서 IPsec처리 부분을 생략하는 경우) 단순히 패킷을 전달하는 형태에서 측정한 성능조차 100Mbps에 도달하지 못한다. 〔그림 4〕은 소프트웨어 방식에서 IP 패킷 전송률을 나타내고 있다.



〔그림 3〕 PCI FE컨트롤러와 내장 FE컨트롤러의 성능 비교



〔그림 4〕 소프트웨어 방식의 구현의 패킷 전송률

〔그림 4〕는 패킷의 크기가 작을수록 전송률이 현저히 떨어짐을 알 수 있다. 패킷 크기와 관계없이 100Mbps를 보장하기 위해서는 2.4GHz이상의 CPU를 사용해야 한다. 여기에 IPsec을 적용하는 경우 IPsec 처리 성능은 일반 패킷 처리 성능의 20~40% 정도를 보이게 되므로 2.4GHz의 CPU를 사용하더라도 충분한 IPsec 성능을 보장하지 못하게 된다.

이와 같이 소프트웨어 방식의 IPsec 구현은 전형적인 메모리/CPU에 대한 병목 현상을 보이며, 100Mbps의 회선 속도를 제공하기 힘들다. 또한 패킷 전송 과정에서 패킷을 처리하기 위한 CPU의 부하로 시스템의 응답성이 떨어지거나 다운되는 현상이 보이기도 한다.〔11〕〔12〕

2. 암호가속기를 활용한 구현

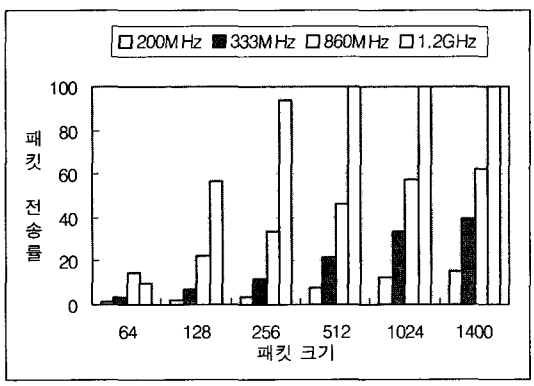
소프트웨어 방식의 성능을 개선하기 위해 일반적으로 IPsec을 위한 암호가속기를 사용하는 것이다. 암호 가속기의 역할은 패킷 데이터를 암호화 혹은 복호화를 하드웨어적으로 수행하는 것이다.

이 방식에서 패킷의 이동경로를 보면 네트워크->FE컨트롤러->메모리->CPU(패킷 읽기)->암호가속기(IPsec 처리)->CPU(패킷 쓰기)->메모리->FE컨트롤러->네트워크이다.

패킷의 이동경로에서 보듯이 CPU가 수행하는 IPsec처리 부분을 암호가속기가 대신하고 있음을 의미한다. 따라서 IPsec처리하는 동안은 CPU가 병렬로 다른 일을 행할 수 있다는 장점을 가지고 있으며 이를 통한 성능의 향상을 꾀할 수 있다. 하지만 이 경우에도 패킷의 전송은 CPU가 담당하고 있어 전체 성능을 향상시키기에는 역부족이다. 즉 암호가속기를 이용하는 경우에도 IPsec 처리 성능은 CPU의 성능과 시스템 컨트롤러의 성능 및 부가적으로 암호가속기의 성능에 영향을 받는다.

〔그림 5〕는 암호 가속기를 이용한 IPsec 구현의 패킷 전송률을 보여주며 〔그림 4〕의 IP 패킷에 대한 전송률과 비교하면 IPSec을 적용하는 경우 패킷 전송율은 1/3 혹은 그 이하로 떨어지게 된다.

암호 가속기와 더불어 사용하는 기타 암호 보조 연산기로는 IP/TCP/UDP 오프로드를 제공하는 것이 있다. IP 패킷을 처리하는 과정에서 많은 시간이 소요되는 기능이 IP 체크섬 및 TCP 혹은 UDP 체크섬을 계산하는 부분이다. 따라서 이러한 기능이 포함된 보조 연산기를 이용하면 추가적인 성능 개선을 보여줄 수 있다. 하지만 IPsec 기능에 있어 주된 성능 저하 요인은 IP 패킷에 대한 처리가 아니라 암호화 연산 과정에 있기 때문에 암호 가속기를 사용하는 경우가 더 좋은 성능을 보인다.〔12〕



〔그림 5〕 암호가속기를 이용한 IPsec 성능

또한 IP/TCP/UDP 오프로드 기능과 암호 가속기를 같이 사용하는 경우에도 물리적인 회선 속도를 제공할 수 없다. 이때는 IP 오프로드 처리 후 시스템 컨트롤러를 경유하여 시스템 메모리로 패킷이 적재되며 CPU는 이 패킷을 확인한 후 다시 시스템 컨트롤러를 경유하여 암호 가속기로 패킷을 전달한다. 암호 가속기에서 처리가 끝난 패킷은 다시 CPU에서 처리한 후 시스템 컨트롤러를 경유하여 IP 오프로드로 전달한다. 이러한 과정에서 CPU 혹은 암호가속기에서 발생하는 병목현상으로 물리적인 회선 속도를 제공하기에는 부족하다.

Ⅲ. SoC 형태로 구현한 FSC2003

IPsec을 고속으로 처리하기 위한 SoC(System on Chip)를 구현하기 위해서는 구성 요소인 네트워크, CPU, IPsec 암호가속기를 단순히 하나로 묶어서는 충분한 성능을 기대할 수 없다.

소프트웨어 방식의 구현에서 검토한 바와 같이 단순한 네트워크 기능을 통합하는 형태가 바로 내장 FE컨트롤러를 사용하는 경우이다. 이 경우 외장 FE컨트롤러(PCI 이더넷 컨트롤러)를 사용하는 경우보다 성능 향상을 기대할 수 있으나, 여전히 상당한 수준의 CPU 성능을 요구한다.

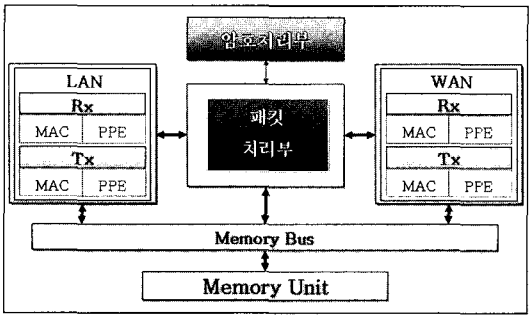
비슷한 방식으로 암호가속기 방식의 구현에서도 IP 오프로드 혹은 암호가속기를 통합하는 경우 역시 CPU를 통한 패킷 처리 과정이 필요하며, 이 또한 상당한 수준의 CPU 성능이 필요하다.

따라서 네트워크, CPU, IPsec 암호가속기를 하나의 SoC로 통합할 때 IPsec 처리가 대부분 CPU의 도움 없이 이루어질 수 있도록 소프트웨어적인 요소를 하드웨어 로직화하는 유기적인 기능 통합이 필요하다. 즉 IP 패킷을 IPsec 패킷으로 변환하는 과정에서 CPU의 관여를 최소화하거나 혹은 완전히 제거함으로써 시스템 컨트롤러를 통한 I/O 과정의 병목현상을 완전히 제거하는 SoC가 요구된다.

IPsec SoC인 FSC2003의 경우 단순히 네트워크, CPU, IPsec의 기능을 통합한 것이 아니라, 각 기능이 상호 연동하여 IP 패킷을 IPsec으로 변환하거나 혹은 반대의 변환 과정에서 CPU의 역할을 최소화할 수 있는 구조를 선택하였다.

이러한 부분을 고려하여 FSC2003은 IP 오프로드 기능을 포함하여 FE컨트롤러를 설계하고, 수신된 IP 패킷을 CPU로 전달하기 전에 패킷을 해석하여 CPU가 추가적인 처리를 해야 하는지 여부를 결정하는 하드웨어 패킷 처리부를 도입하였다. 패킷 처리부에서

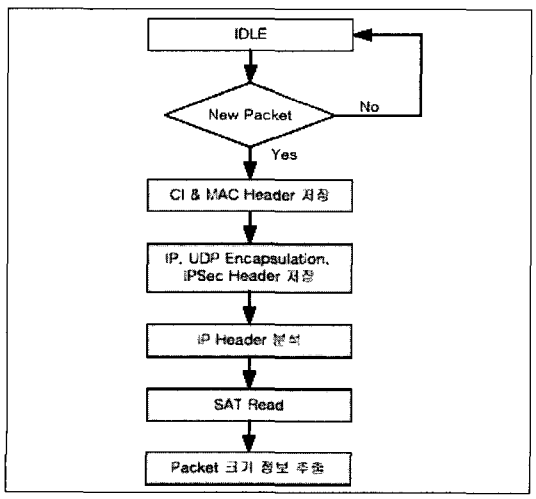
처리할 수 있는 패킷은 바로 하드웨어 암호처리부로 전달하여 IP 패킷을 IPsec 패킷으로 변환한 후 곧바로 FE컨트롤러로 전송하게 된다.



〔그림 6〕 FSC2003의 패킷 흐름도

〔그림 6〕과 같이 패킷 처리부는 네트워크 세션 처리 및 IPsec SA(Security Association)처리를 담당하게 된다. IPsec 기능은 호스트 기반 혹은 세션 기반으로 동작하도록 설계할 수 있으나, 유연성과 응용성을 높이기 위해서는 세션 기반 IPsec 패킷 처리 기능이 필요하다.

〔그림 7〕과 같이 패킷 처리부는 새로운 패킷이 수신되는 경우 이더넷 헤더 및 IP 헤더를 분석하고 오류 여부를 판단하여 잘못된 패킷은 버리며, 정상적인 패킷에 대하여 SA 테이블 검색을 시도한다. SA 테이블이 존재하는 경우 해당 SA 정보와 패킷의 부가적인 정보들을 추출한 후 암호화 혹은 복호화를 위하여 암호 처리부로 패킷을 전달한다.

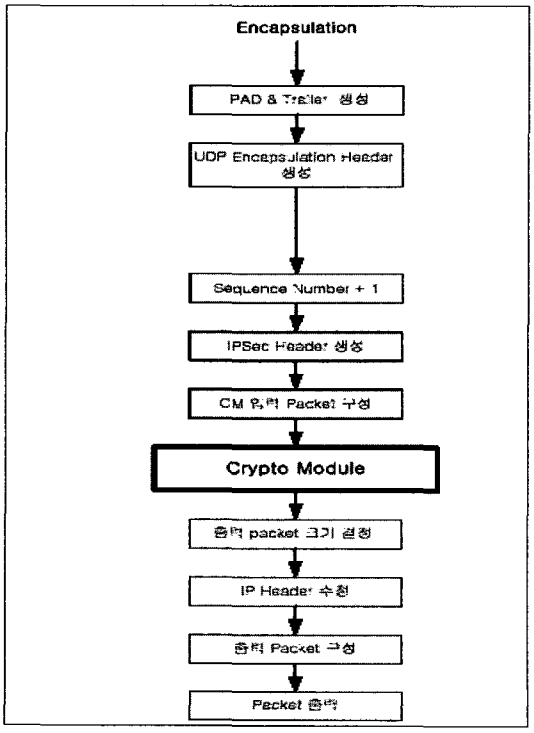


〔그림 7〕 패킷 처리부 하드웨어 로직의 동작

패킷의 암호화의 경우 〔그림 8〕과 같이 암호 처리부는 패킷 처리부가 전달한 IP 패킷에 대하여 패딩 정보와 IPsec 헤더를 생성한다. 그리고 IPsec 터널 처리를 위하여 IP 헤더를 생성하고 암호화를 위해 크립토 모듈로 전달할 정보를 생성한다. 크립토 모듈에서 처리가 완료되면 암호화된 패킷을 참조하여 외부 IP 헤더에 포함된 패킷의 길이 등을 조정한 후 다시 패킷 처리부로 전달하여 암호화된 패킷이 전송될 수 있도록 한다.

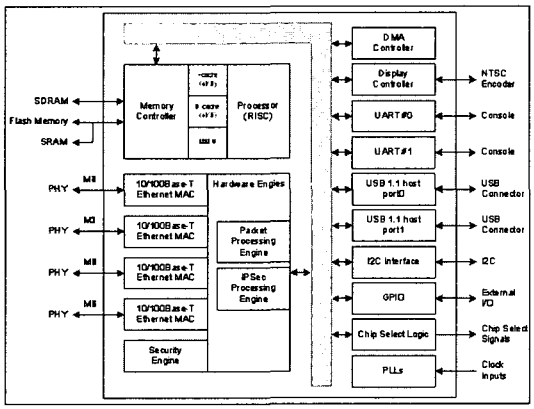
IPsec SoC인 FSC2003의 특징은 패킷을 전달할 때 IPsec SA가 형성된 경우 CPU의 개입 없이 하드웨어 로직이 패킷을 처리하도록 설계된 것이 특징이다. CPU는 IPsec SA가 존재하지 않는 경우 SA를 만드는 과정에만 개입하게 된다. 이 결과 인터페이스 상에서 생기는 오버헤드를 완전히 없애 고속의 IPsec 처리가 가능하도록 했다.

FSC2003의 패킷 이동경로는 SA가 존재하는 경우 네트워크->FE컨트롤러->메모리->IPsec엔진->메모리->FE컨트롤러->네트워크이며 SA가 존재하지 않는 경우 네트워크->FE컨트롤러->메모리->CPU(SA 형성)->메모리->FE컨트롤러->네트워크이다.



〔그림 8〕 암호 처리부 하드웨어 로직의 동작

CPU의 관여는 세션이 처음 만들어지는 시기로 SA를 형성하기 위해서이며 SA 형성 후 패킷의 전달은 하드웨어 로직에 의해 이루어지므로 100Mbps의 부하가 걸려도 CPU의 사용량은 거의 0%에 가깝게 유지된다.



〔그림 9〕 FSC2003 SoC 구조

FSC2003은 IPsec처리 기능 외에 NAT, PPPoE, 방화벽 기능을 하드웨어 로직으로 처리하며 2x2 IPsec FE 스위치 및 USB1.1, I2C, RS232C, NTSC 인코더 등 다양한 주변장치를 제어할 수 있는 인터페이스를 가지고 있다.

0.18u 반도체 공정을 사용하여 개발되었으며 패키지는 324핀 TEBGA 타입이다.

Ⅳ. FSC2003의 성능

다음은 실제 FSC2003의 성능 결과이다. 성능은 IP 패킷과 IPsec 패킷의 Throughput으로 측정한다.〔10〕

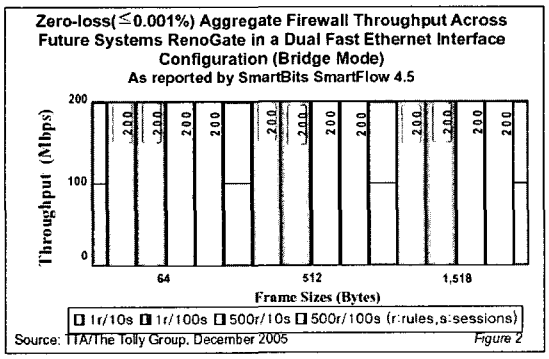
〔그림 10〕와 같이 실제 제작된 장비에서 200Mbps의 성능을 제공하고 있으며, 시험 장비를 통한 검증에서는 최대 400Mbps의 성능을 제공한다.

〔표 2〕 SoC를 이용한 IPsec 성능 결과(ESP with ICV, ICV 알고리즘은 SHA1으로 고정). IP 패킷을 기준으로 한 throughput 결과.

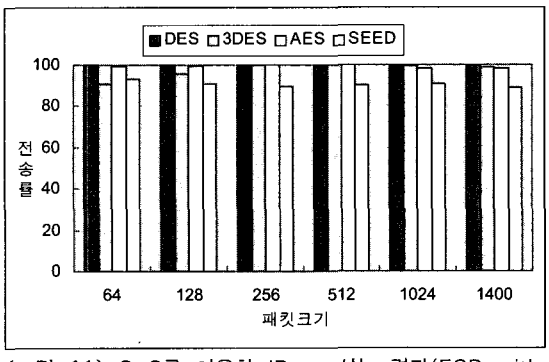
IP패킷 크기	IP Throughput			
	DES	3DES	AES	SEED
64	62.69	56.95	58.95	55.08
128	74.56	71.50	71.50	65.20
256	84.53	84.53	82.51	73.89
512	91.33	91.33	90.02	81.29
1024	95.00	95.00	93.13	85.68
1400	96.30	95.30	95.01	85.64

〔표 1〕에서 정리한 이론적인 최대 IPsec 성능을 참고하여 〔표 2〕의 Throughput을 비교하면 DES 알고리즘의 경우 이론적인 최대 성능을 보이며, 3DES는 256바이트부터 이론적인 최대 성능이 제공된다. 알고리즘에 따른 성능의 차이는 암호 알고리즘을 계산하기 위한 시간이 알고리즘별로 차이가 있기 때문이다.

〔그림 11〕은 〔표 1〕을 참조하여 〔표 2〕의 IP 패킷을 기준으로 한 전송률에서 IPsec 패킷을 기준으로 한 전송률로 변환한 것이다. DES 및 AES 알고리즘을 사용하는 경우 100Mbps FE 회선속도로 IPsec 패킷을 처리한다. 3DES는 64바이트에서 90Mbps



〔그림 10〕 FSC2003 SoC를 이용한 IP 패킷의 Throughput



〔그림 11〕 SoC를 이용한 IPsec 성능 결과(ESP with ICV, ICV 알고리즘은 SHA1으로 고정). IPsec 패킷을 기준으로 한 throughput 결과.

의 성능을 보이고 256바이트에서부터는 회선속도를 제공한다. SEED 알고리즘의 경우 평균적으로 90Mbps의 성능을 보인다.

V. 결 론

IPsec 구현 방법인 소프트웨어 방식, 암호가속기 연동 방식, SoC 구현 방식 등 세 가지 형태 중 SoC 구현 방식이 가장 좋은 성능을 보이며, SoC 구현 방식도 IP 패킷을 IPsec 패킷으로 변환하거나 혹은 반대의 변환에서 하드웨어를 이용하여 CPU 개입 없이 자동 변환하도록 구성하는 경우 가장 좋은 성능을 보인다.

100Mbps FE의 회선 속도를 제공하기 위해서는 소프트웨어의 경우 2.4GHz이상의 CPU를 사용해야 IP 패킷에 대한 회선 속도를 제공할 수 있으며, IPsec 암호가속기를 사용하는 경우에도 여전히 회선 속도를 제공하지 못한다. IPsec SoC인 FSC2003을 이용하는 경우에 비해서 소프트웨어 방식은 약 100배 이상의 비용을 들여야 하며 암호가속기를 이용하는 경우에도 50배 이상의 비용을 들여야 동일한 성능을 얻을 수 있다.

따라서 IPsec 처리 SoC를 이용하는 경우에 비용적인 측면에서 다른 방식과는 비교할 수 없는 성능을 보이게 된다.

참 고 문 헌

[1] J. Postel, "INTERNET PROTOCOL", RFC791, September 1981.

[2] Digital Equipment Corporation, Intel Corporation, Xerox Corporation, "The Ethernet: A Local Area Network", September 1980.

[3] S. Kent, "IP Authentication Header", RFC4302, December 2005.

[4] S. Kent, "IP Encapsulating Security Payload(ESP)", RFC4303, December 2005.

[5] R. Pereira, R. Adams, "The ESP CBC-Mode Cipher Algorithms", RFC2451, November 1998

[6] H.J. Lee, J.H. Yoon, S.L. Lee, J.I. Lee, "The SEED Cipher Algorithm and Its Use with IPsec", RFC4196, October 2005

[7] S. Frankel, R. Glenn, S. Kelly, "The AES-CBC Cipher Algorithm and Its Use with IPsec", RFC3602, September 2003.

[8] S. Bradner, J. Mcquaid, "Benchmarking Methodology for Network Interconnect Devices", RFC2544, March 1999.

[9] Sunil Kalidindi, Elliott Stewart, "IPsec Virtual Private Networks: Conformance and Performance Testing", IXIA Corporation, White Paper.

[10] Tolly Group, "Future Systems, Inc. FSC2003 SoC(System on a Chip) in Future Systems RenoGate, Firewall and VPN Performance Evaluation", January 2006.

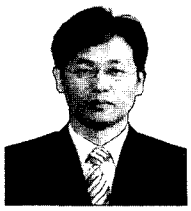
[11] Annie P. Foong, Thomas R. Huff, Herbert H. Hum, Jaidev P. Patwardhan, Greg J. Regnier, "TCP Performance Re-Visited", Interl Corporation, March 2003.

[12] Andy Currid, "TCP Offload to the Rescue", ACM Queue vol 2, no. 3, May 2004

[13] J. Wu, A. Mamidala, D. K. Panda, "Can NIC Memory in InfiniBand Benefit Communication Performance? - A Study with Mellanox Adapter", April, 2004

〈著 者 紹 介〉

정 영 조 (Y.C. Chung)



1985년 2월 : 홍익대학교 학사
1987년 2월 : KAIST 석사
1993년 3월~현재 : (주) 퓨쳐시스템 홈네트워크 사업 본부장
관심분야 : 정보보호, 홈네트워크

김 종 혁 (John Kim)

1993년 2월 : KAIST 학사 졸업
1995년 2월 : KAIST 석사 졸업
1995년 3월~현재 : (주) 퓨쳐시스템
관심분야 : 통신공학, 정보보호, 홈 네트워크

조 인 현 (InHyun Cho)

1996년 2월 : 동아대학교 학사
1998년 2월 : 동아대학교 석사
1998년 3월~현재 : (주) 퓨쳐시스템
관심분야 : 전자공학, 통신공학, 정보보호

김 현 철 (KimHyunChul)

1994년 2월 : 대불대학교 학사
1998년 2월 : 전남대학교 석사
2000년 6월~현재 : (주) 퓨쳐시스템
관심분야 : 통신공학, 근거리무선통신