Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.24, NO.6, Dec. 2014 ISSN 2288-2715(Online)
http://dx.doi.org/10.13089/JKIISC.2014.24.6.1065

AES-NI를 이용한 VPN 암호화 가속화*

정 진 표,[1†] 황 준 호,[3] 한 근 희,[2‡] 김 석 우[3]
[1]성균관대학교, [2]고려대학교, [3]한세대학교

Accelerated VPN Encryption using AES-NI*

Jin-pyo Jeong,[1†] Jun-ho Hwang,[3] Keun-hee Han,[2‡] Seok-woo Kim[3]
[1]Sungkyungkwan University, [2]Korea University, [3]Hansei University

요 약

IPSec 기반의 VPN에서는 데이터의 암호화 안전성 및 성능을 고려하였을 때 대칭키 기반의 AES 알고리즘의 성능이 가장 우수하다고 할 수 있다. 하지만 IPSec 기반 VPN에서 AES 알고리즘을 사용할 때 VPN의 성능은Cavium Networks사의 OCTEON Card 시리즈 같은 고가의 하드웨어 암호화 가속화 카드를 사용해도 동일한 하드웨어를 사용하는 방화벽의 절반의 성능도 내지 못하는 것을 알 수 있다. 2008년 인텔에서는 인텔 CPU에서 AES 알고리즘의 성능을 향상시키기 위해 AES-NI 7개의 명령어 집합을 발표하였다. 본 논문에서는 인텔 CPU의 AES-NI 7개의 명령어 집합을 사용 할 경우 IPSec 기반 VPN에서 실제로 성능이 얼마나 향상되는 지 검증 한다.

ABSTRACT

Considering the safety of the data and performance, it can be said that the performance of the AES algorithm in a symmetric key-based encryption is the best in the IPSec-based VPN. When using the AES algorithm in IPSec-based VPN even with the expensive hardware encryption card such as OCTEON Card series of Cavium Networks, the Performance of VPN works less than half of the firewall using the same hardware. In 2008, Intel announced a set of 7 AES-NI instructions in order to improve the performance of the AES algorithm on the Intel CPU. In this paper, we verify how much the performance IPSec-based VPN can be improved when using seven sets of AES-NI instruction of the Intel CPU.

Keywords: VPN, IPSec, AES, AES-NI

I. 서 론

IP Security(이하 IPSec) 기반 VPN에서는 ESP 암복호화를 위해 일반적으로 3DES, AES, AIRA, SEED 알고리즘을 이용하며, 이는 모두 대칭키 기반의 알고리즘으로 128bit 이상의 키를 사용한다. 하지만 3DES는 56bit 키를 사용하는 DES 알고리즘을 3회 반복하여 키의 길이를 168bit로 확장하였을 뿐 블록 크기는 여전히 64bit이며, DES 연산이 3회 반복되기 때문에 성능 저하가 발생한다. 또한 1999년 RSA DES Challenge에서 취약성이 증명됨에 따라 NIST에서는 2001년 발표한 AES 알고리즘의 사용을 권고하고 있다. AES 알고리즘은 국제표준 알고리즘으로 채택되었으며, 128bit 블록 크기와 128, 192. 256bit 길이의 키를 사용한다. 국내에서도 1999년 2월 민간분야에서 전자상거래, 금융권, 무선통신 등에서 전송되는 중요 정보를 보호하기 위해 KISA를 중심으로 개발된 SEED(128, 256bit 키) 알고리즘과 2004년 국가정보원을 중심산,학,연 공동

접수일(2014년 9월 17일), 수정일(1차: 2014년 10월 20일, 2차: 2014년 10월 30일), 게재확정일(2014년 11월 13일)
* 본 연구는 미래창조과학부 및 정보통신산업진흥원의 대학 IT연구센터육성 지원 사업/IT융합고급인력과정지원사업 (NIPA-2014-H0301-14-102) 연구결과로 수행하였습니다.
† 주저자, jinpyoj@gmail.com
‡ 교신저자, khhan@formal.korea.ac.kr (Corresponding author)

으로 AES와 동일한 규격의 128bit 블록을 이용하는 ARIA(128, 192, 256bit 키) 알고리즘이 개발되었으며, 공공기관에 도입되는 보안제품에 탑재하는 것을 의무화하고 있다. IPSec 기반의 VPN에서는 데이터의 암·복호화로 인한 오버헤드가 발생하기 때문에 국내 보안업체에서는 VPN의 성능을 보장하기 위해 Cavium Networks와 같은 외산 전용 암호화 가속화 카드에 의존하고 있지만 암호화 가속화 카드를 장착한 국내 보안 업체의 VPN 제품도 가용회선 대비 50% 미만의 성능을 보여주고 있다[Table 1].

대다수 국내 보안 업체에서 이용하는 암호화 가속화 카드의 경우 3DES, AES와 같은 국제 표준 암호 알고리즘만을 지원하고 있으며, 국산 알고리즘은 현재 지원하지 않는 상황이다. SEED의 경우 2005년 IETF 국제 표준으로 지정됨에 따라 추후 외산 암호화 가속화 카드 생산 업체에서 지원 받을 수 있으나, 국내 공공기관에서 사용이 의무화되고 있는 ARIA 알고리즘의 경우에는 암호화 카드의 지원을 받을 수 없는 실정이며, 국내에서 개발된 ARIA 가속화(OPERA)칩에서는 인터페이스의 한계(현재 PCI 인터페이스 미 지원)로 VPN에서의 성능 향상을 기대하기 힘든 실정이다. 이러한 이유로 현재 국내 보안 업체에서 생산하는 VPN 장비의 경우 대다수 전용 암호화 가속화 카드의 지원을 받아 AES 알고리즘 기반의 암·복호화를 사용하고 있다. Intel에서는 2008년 이후 출시되는 CPU에 6개의 AES 관련 명령어 집합(New-Instruction)을 포함하였는데[1], 별도의 하드웨어에 의존하지 않은 AES 암·복호화 성능 향상을 주장하고 있다. 2010년 7번째 명령어인 PCLMULQDQ이 추가되며, 암·복호화 성능에 의한 부하가 큰 IPSec 기반의 VPN에서 블록 암호 운영 모드 중 하나인 GCM을 권장하고 있다[2,3]. AES-NI(Advanced Encryption Standard-New Instructions)는 가장 최근 출시된 4세대 Intel CPU에서 AES의 암·복호화 성능을 더욱 향상시킨 것으로, 범용적인 분야에 적용할 수 있도록 지원하고 있다[4].

Table 1. VPN Performance

Ma	Model	CPU Core	Mem GB	Max (bps)	VPN (bps)
A	31	1	1	1G	500M
	50	1	2	1.5G	150M
	70	2	2	2G	180M
	100	2	4	4G	500M
	400	2	4	6G	700M
	500	4	8	8G	900M
	1000	6	8	12G	3G
	5000	8	16	30G	4G
	10000	12	16	50G	8G
	22000	16	28	100G	8G
S	100	1	2	500M	300M
	300	2	4	1G	300M
	500	2	4	2G	700M
	1000	2	4	4G	1.5G
	1500	4	8	6G	1.5G
	2000	4	8	10G	2G
	3000	6	12	20G	3G
	6000	6(x2)	24	40G	5.5G
F	365	1	512M	500M	200M
	500	2	1	1G	500M
	1000	2	1	2G	700M
	1000F	2	1	2G	700M
	2000	4	2	2G	1G
	4000	4	2	4G	1.5G
	6500	4	4	10G	2G
	8000	4(x2)	8	20G	4G
	10000	6(x2)	12	50G	6G

본 논문에서는 IPSec 기반 VPN의 ESP 패킷 암·복호화에 Intel 칩의 AES-NI의 7가지 명령어 집합을 이용하였으며, AES 운영 모드 중 GCM을 사용하여 VPN의 대역폭 성능 향상을 증명하고자 한다. 증명을 위해 실제 인텔 AES-NI가 적용된 기가급 VPN을 구현하여 ESP 패킷의 암·복호화를 IXIA 계측 장비를 이용하여 가용 회선 대역폭 성능을 측정하였다. 본 논문의 구성은 다음과 같다. Ⅱ장에서는 관련 연구를 설명하고, Ⅲ장은 구현 환경 및 성능 평가를 실시하여 Ⅳ장에서 측정 결과 분석을 통하여 각 VPN 제조사에서 암호화 가속화 카드 없이 AES-NI를 활용하여 성능개선을 해 볼 것을 제안하고, 마지막으로 Ⅴ장에서 본 논문이 제안하는 결론을 제안한다.

II. 관련 연구

2.1 IPSec

IPSec은 구현 방법에 따라 “End to End”, “VPN”, “Road Warrior”, “Nested tunnel”로 이용이 가능하다. End to End는 서로 통신하는 두 개의 호스트(이하 Host)가 서로 IPSec 모듈을 가지고 있어 다른 장비의 도움이 없이 IPSec Data를 주고받는 형태이다. VPN은 Host 입장에서는 자신의 데이터 암호화 유무를 알지 못 하며, Gateway(이하

G/W)에 의해 IPSec 데이터를 주고받는 형태이다. Road Warrior는 이동 중에 있는 Host와 G/W 간에 IPSec 데이터를 주고받는 형태이다. Nested Tunnel은 2개의 G/W를 이용하여 IPSec 데이터를 이중 암호화하는 형태이다.

IPSec은 IKE(Internet Key Exchange)를 통한 SA(Security Association, 이하 보안 연계)를 구성한 후 ESP(Encapsulating Security Payload) 또는 AH(Authentication Header) 프로토콜을 이용하여 IP Packet의 보안 및 인증, 무결성, 기밀성을 제공한다. 2005년 NIST에서 IPSec 기반의 VPN에서 기밀성을 위해 AESCBC와 무결성을 위해 HMAC-SHA1을 권고하고 있다. 인증 및 암호화 알고리즘 및 키 정보 등을 포함하는 보안 연계의 유효기간을 IPSec에서는 최대 8시간으로 권고하고 있으며, IKEv1(IKE Version 1), IKEv2(IKE Version 2)에서는 최대 24시간 이내에 갱신 할 것을 권고하고 있다.

2.1.1 IKE

IKE는 ISAKMP (Internet Security Association and Key Management Protocol)와 Oakley Key Determination Protocol, SKEME(a versatile Secure Key Exchange Mechanism for internet)으로 총 3가지 프로토콜이 결합 된 IPSec의 보안 연계와 키 관리 프로토콜이다. ISAKMP도 키 교환 프로토콜로 사용되지만 키 교환을 위한 프레임 환경을 제공해줄 뿐 실질적인 키 교환 방식을 제안하지 않는다는 점에서 IKE와 다르며, IKE는 ISAKMP의 키 교환 프레임 환경을 이용하여 키를 교환한다.

2.1.2 AH

AH는 데이터의 무결성과 IP 패킷의 인증, 재전송 공격 방지 서비스를 제공하며, 패킷의 헤더를 포함하여 인증하며, 대칭키 기반의 알고리즘을 사용한다.

전송 모드(Transport Mode)와 터널 모드(Tunnel Mode) 2가지 모드로 사용할 수 있다[Fig. 1].

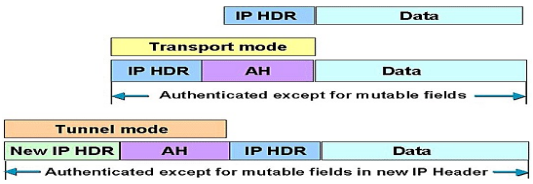


Fig. 1. AH

2.1.3 ESP

ESP 프로토콜은 AH 프로토콜과 달리 데이터의 헤더를 제외하고 인증하며, 데이터 암호화 기능이 제공된다. ESP 뒤의 데이터는 암호화되어 전송되며, IKE에 의해 미리 교환된 키를 이용하여 수신측에서 데이터를 복호화하게 된다.

ESP는 전송 모드와 터널 모드 2가지 모드로 사용할 수 있다[Fig. 1].

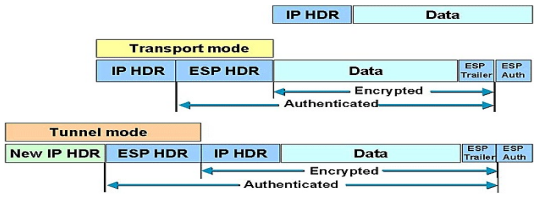


Fig. 2. ESP

2.2 AES-NI

AES-NI 명령어는 암호화를 위해 AES 알고리즘을 사용하는 모든 응용 프로그램에서 사용할 수 있다. AES 알고리즘은 네트워크 암호화, 디스크 및 파일 암호화와 같은 여러 응용 프로그램에서 사용되고 있다. 파일 및 디스크 암호화에서는 디스크에 저장된 데이터를 보호하기 위해 AES를 사용하며, 네트워크 응용 프로그램은 SSL, TLS, IPSec, https, ftp, ssh 등 다양한 프로토콜의 데이터를 보호하기 위해 사용한다. 4세대 CPU(Hasewll)에서는 총 7개의 명령어 집합[Table 2]을 제공한다.

Table 2. AES-NI Instructions

Instructions	sorting
AESENC	encryption
AESENCLAST	encryption
AESDEC	decipher
AESDECLAST	decipher
AESKEYGENASSIST	key generate
AESIMC	key convert
PCLMULQDQ	Improving processing operations that carry any

AES-NI의 6가지 명령어 집합(PCLMULQDQ 제외)을 이용하여 IPSec의 데이터를 암·복호화 할 경우 약 50%의 CPU 오버헤드가 감소되었다[5]. PCLMULQDQ 명령어는 암·복호화와 인증을 동시에 수행하는 GCM(Galois/Counter Mode)을 사용한 IPSec에서 동일 AES-NI의 지원을 받지 않은 플랫폼 대비 400%의 처리량을 향상시켰다[2,3].

Linux cryptographic framework 기반의 AES-NI는 [Fig. 2]과 같이 구성된다.

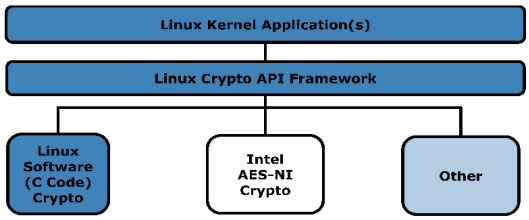


Fig. 3. Linux cryptographic framework

2.2.1 AES-NI 지원

AES-NI 명령어 집합은 BIOS와 OS(이하 운영체제)에서 지원 유무 확인 및 동작 여부를 설정할 수 있다[Fig. 4].

현재 동작되는 운영 체제의 Kernel에서 인텔이 제공하는 Driver를 포함 한 경우 [Fig. 5]와 같이

Fig. 4. Check support AES-NI on BIOS

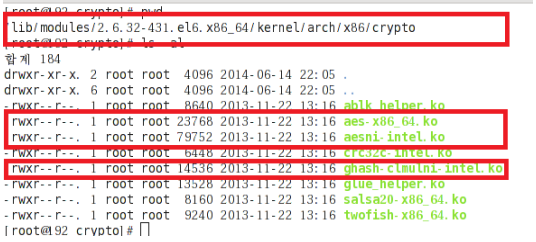


Fig. 5. AES-NI Support (Kernel-2.6)

AES-NI의 지원 유무를 확인할 수 있다.

Linux Kernel-2.6.32 이상부터는 BIOS에서 AES-NI 동작 설정 유무에 따라 운영 체제 부팅 후 AES-NI를 자동으로 지원해주고 있다. [Fig. 6]과 같이 BIOS에서 AES-NI의 동작을 설정 하였을 경우 정상적으로 동작됨을 확인 할 수 있다.

또한 Library나 Crypto API, 응용프로그램 등 다양한 방식을 통해 동작되며 점차 사용 범위가 확대되고 있다[4]. 하지만 BIOS에서 AES-NI 지원을 ON/OFF 하는 것만으로는 CBC 모드만을 지원하기 때문에[Table 2] PCLMULQDQ 명령어 지원을 받지 못하기 때문에 인텔에서 권장하는 최적의 성능을 사용하기 위해서는 AES-NI를 사용하고자 하는 대상의 별도의 환경 구현이 필요하다.

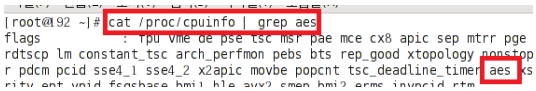


Fig. 6. Check support AES-NI

2.2.2 AES-NI 명령어 집합

AESENC와 AESENCLAST는 AES 암호화 라운드를 대상으로 하는 명령어이다. AESENC는 암호화의 단일 라운드를 수행한다. 하나의 명령으로 ShiftRows, SubBytes, MixColumns, Add RoundKey의 네 단계를 결합하여 수행한다.

AESDEC와 AESDECLAST는 등가 역 암호를 이용하여 AES 복호화 라운드를 대상으로 하는 명령어이다. AESDEC는 복호화의 단일 라운드를 수행하여, InvShiftRows, InvSubBytes, InvMixColumns, AddRoundKey의 네 단계를 결합하여 수행한다.

AESIMC는 등가 역 암호를 사용하여 암호 해독에 사용할 수 있는 형태로 암호화 라운드 키를 변환하는데 사용한다.

AESKEYGENASSIST는 암호화에 사용되는 라운드 키를 생성하는데 사용된다.

PCLMULQDQ는 128bit 결과에 두 64bit 데이터의 자리 올림이 없는 곱셈을 수행한다. 256bit 결과에 두 128bit 데이터의 자리 올림 없는 곱셈은 빌딩 블록으로 PCLMULQDQ를 사용할 수 있다. 자리올림 없는 곱셈은 블록 암호인 Galois/Counter Mode(GCM) 동작을 구현하는 중요한 부분이다.

GCM 동작은 인증 기능을 추가하기 위해 AES 알고리즘과 함께 사용될 수 있다. 또한 임의 다항식을 이용한 해시 함수와 CRC 계산에 사용될 수 있다.

〔Fig. 7〕과 같이 AES 운영 모드 중 GCM 성능 향상을 지원하는 "PCLMULQDQ"의 지원 유무를 확인해볼 수 있다.

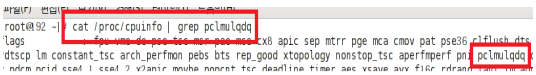


Fig. 7. Check support PCLMULQDQ

2.3 블록 암호 알고리즘의 운영 모드

본 논문의 성능 측정 대상이 되는 IPSec 기반의 VPN에서는 데이터의 기밀성을 위해 대칭키 기반의 블록 알고리즘을 사용하며, 무결성 검증을 위해 MAC(Message Authentication Code)을 사용하였다. DES에서 고안된 운영 모드는 대칭키 기반 블록 암호화 알고리즘에 모두 적용할 수 있으며, 운영 모드에 따라 성능, 안전성뿐만 아니라 데이터의 기밀성과 무결성 제공 유무의 차이를 보인다. 운영 모드 중 기밀성과 무결성을 동시에 제공하는 CCM, GCM은 CMVP 인증의 검증대상 보호함수로써〔8〕 암호화 연산으로 인한 오버헤드를 줄이기 위해 연산 처리능력이 민감한 임베디드 장비에서 많이 사용되고 있다〔8〕. AES-NI 명령어 중 "PCLMULQDQ" 는 GCM의 메시지 인증 처리 방식인 자리 올림 없는 곱셈에 최적화 된 명령어로써 AESGCM을 사용한 IPSec에서 AES-NI의 지원을 받지 않은 동일 플랫폼 대비 400%의 처리량을 향상시켰다〔2,3〕. 본 논문에서는 IPSec 기반의 VPN에서 IKE, ESP 패킷이 필요로 하는 암·복호화 및 인증에서 가장 우수한 성능의 측정을 위해 AES의 4가지 운영 모드(CBC, CTR, CCM, GCM)를 사용하였다.

2.3.1 CBC(Cipher Block Chaining)

CBC 모드는 AES 블록 암호화 운영 모드 중 가장 많이 사용되는 모드로 2005년 NIST에서 IPSec 기반의 VPN에서 기밀성을 위해 AESCBC를 권고하고 있다. 〔Fig. 8〕은 CBC 암호화 과정으로 평문의 각 블록은 XOR 연산을 통해 이전 암호문과 연상되고, 첫 번째 암호문에 대해서는 IV(Intial Vector)가 암호문 대신 사용되며, 동일한 평문이 동일한 암호문이

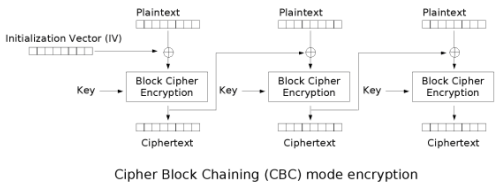


Fig. 8. CBC Mode

되지 않는다. IV는 제 2의 키가 될 수 있다. CBC 모드의 특징은 블록의 연산이 순차적으로 수행되며, 암호문이 블록의 배수가 되기 때문에 복호화 후 평문을 얻기 위해 Padding을 해야만 한다. 암호화가 병렬처리가 아닌 순차적으로 수행되어야하기 때문에 병렬처리가 불가능하며, 한 블록의 bit 연산 오류가 발생할 경우 순차적인 블록에 영향을 미치게 된다.

2.3.2 CTR(CounTeR)

블록 암호를 스트림 암호처럼 사용하기 위해 고안되었다. 블록을 암호화 할 때마다 1씩 증가하는 IV(Nonce)와 Counter를 암호화해서 키 스트림을 만든다. 즉, IV와 Counter를 암호화한 비트열과 평문 블록과의 XOR을 취한 결과가 암호문 블록이 된다.

스트림 암호의 일종으로 암호화와 복호화는 완전히 같은 구조가 되므로 구현이 간단하다. 비밀 키와 IV가 주어지면 암호문을 생성할 수 있어 병렬처리가 가능하며, 블록의 순서를 임의로 암호화 및 복호화 할 수 있다. CBC Mode에 비해 성능이 빠르단 장점이 있지만, 동일한 비밀 키와 IV를 반복하여 사용할 경우 안전성에 문제가 생기는 취약점이 있다.

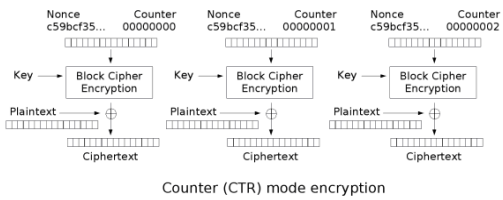


Fig. 9. CTR Mode

2.3.3 CCM(Counter with Cbc-Mac)

CTR 모드의 변종으로 블록 암호를 기반으로 하여 메시지의 기밀성 및 무결성을 동시에 제공하는 블록암

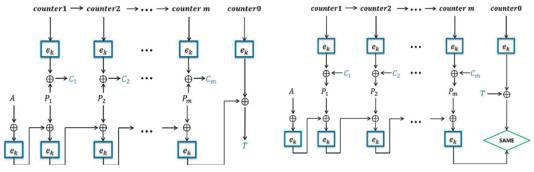


Fig. 10. CCM

호 운영모드이다. 암·복호화와 인증을 동시에 제공하며, CTR Mode를 이용하여 암·복호화 하며, CBC-MAC(Cipher Block Chaining-Message Authentication Code)을 이용하여 인증을 제공한다.

CBC-MAC은 송신측과 수신측에 비밀 키를 공유함으로써 전송된 정보가 변경되지 않고 본래의 정보임을 보증하는 방식이다. 해쉬 기반의 HMAC과 달리 블록 암호를 이용하며, Key는 블록 암호화키로 사용하여 CBC 암호화 모드로 메시지를 암호화하고 일부를 MAC으로 사용한다.

첫 번째 암호화 될 문자로 암호화할 때에는 이전의 암호 문자가 존재하지 않으므로, 초기 벡터(IV: Initial Vector)로 불리는 초기 값을 사용한다. 인증값 생성 시 블록암호의 출력 값을 이용한다. CCM을 이용하여 IPSec을 구현 할 경우 자체적으로 CBC-MAC 해쉬 알고리즘을 이용하여 무결성 테스트를 진행하게 된다. CCM에서는 메시지 인증코드로 블록기반 해시 함수인 CMAC(Cipher-MAC)을 사용하며, 8-octet(64 bit), 12-octet(96 bit), 16-octet(128 bit)의 3가지 ICV를 제공한다.

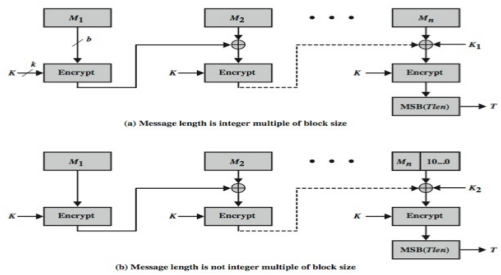


Fig. 11. CBC-MAC

2.3.4 GCM(Galois/Counter Mode)

평문 메시지는 CTR의 변종에 의하여 암호화하며 적은 비용과 적은 대기 시간으로 많은 처리량을 제공하도록 병렬 처리 설계한 방식이다.

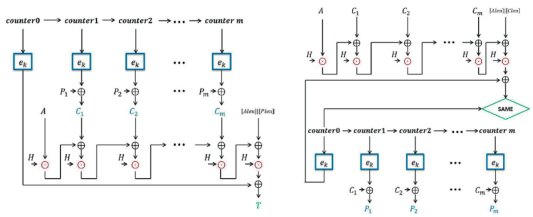
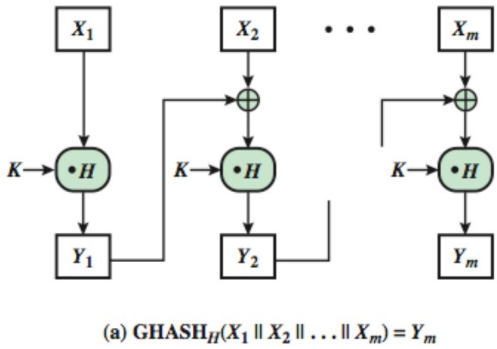


Fig. 12. GCM

이진 Galois 체상에서 규정된 GHash 함수를 이용하여 기밀 데이터의 인증을 보장하는 운영 모드로서 메시지의 기밀성과 인증 기능을 동시에 제공한다. 비밀 키 K와 임의 길이의 비트 문자열 X를 입력으로 하여 비트 길이가 len(X)인 암호문 Y를 반환한다.

유한체 곱 연산을 이용하여 메시지 인증 과정이 포함되어있다. GCM에서는 메시지 인증코드로 블록기반 해시 함수인 GMAC(Galois-MAC)을 사용하며, 8-octet(64 bit), 12-octet(96 bit), 16-octet (128 bit)의 3가지 ICV를 제공한다.



(a) $\mathbf{GHASH}_H(X_1 \| X_2 \| \ldots \| X_m) = Y_m$

Fig. 13. GHASH

III. 환경 구현 및 성능 측정

3.1 구현 환경

성능 측정을 위한 G/W의 사양은 〔Table 3〕과 같다.

G/W 성능은 객관적인 성능 비교를 위해 〔Table 1〕 중 S사의 300 모델을 기준으로 하였다. S사의 300 모델은 2개의 CPU 코어와 4G 메모리를 장착하였으며, 1G의 가용 회선 대비 300Mbps VPN 성능을 보이고 있다. 본 논문에서는 해당 모델과 동일하게 구축 및 성능을 테스트 하기 위해 G/W는

Table 3. G/W Spec

H/W	CPU	Intel Core i5 3.2GHz (Hasewll)
	Mem.	4GB DDR3
	MainBoard	MSi B85M-E45
	NIC	1Gbps
	UTP	Cat5/Cat5e
S/W	OS	CentOS 6.5 Final
	Kernel	2.6.32
	VPN	StrongSwan Developer Release 5

BIOS 환경 설정을 통해 CPU의 총 4개 코어 중 2개만 동작하도록 설정하였고, 4G 메모리를 장착하였다. 또한 1G급 NIC 카드를 장착하여, UTP 카테고리 중 Cat5e를 사용하여 1Gbps급 가용회선을 확인하였다. 국내 보안 업체에서 VPN의 성능 측정을 위해 사용하는 IXIA 계측 장비를 동일하게 사용하여 성능을 측정하였다.

〔Fig. 14〕와 같이 IPSec이 동작되고 있는 G/W1과 G/W2는 IXIA 장비 사이의 패킷을 암·복호화 하게 된다. 트래픽 생성 시 부하가 성능 측정에 영향을 미칠 수 있기 때문에 G/W에서 직접 트래픽을 생성하지 않고, IXIA의 패킷을 암·복호화 하는 VPN의 역할만 수행한다.

AES-NI의 IPSec 성능 향상 구현을 위해 인텔에서 제안하는 IPSec 기반의 AES-NI 성능 측정 문서〔2〕에 기반을 두어 성능에 최적화 된 BIOS를 설정하였다. G/W의 BIOS 설정은 〔Table 4〕와 같다.

AES-NI의 동작 유무는 성능 측정 상황에 따라 설정 값을 변경하여 측정하였다.

C-State는 CPU의 전력을 제어되는 2가지 상태

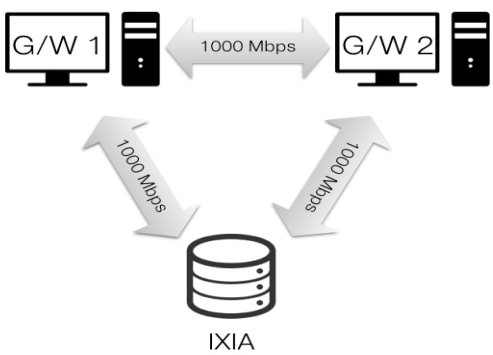


Fig. 14. Network configuration

Table 4. BIOS setting

Menu	set-up
AES-NI	YES
Core	2
C-State	NO
SpeedStep	NO
Cache & Hardware Prefetchers	YES
Hyper-Threading	unsupported

값(C-State, P-state) 중 한가지로 인텔에서 제공하는 전원 관리 시스템이다. 운영 상태(operating state)의 제어로 장시간 작업이 이뤄지지 않을 경우 프로세서의 기능을 낮추어 C0~C7과 같이 단계별 차이를 두어 전력 소비량을 줄이는 방식이다.

Speedstep은 소프트웨어 기반의 프로세서의 클럭(clock) 속도를 변경할 수 있도록 해주며, 전력 결함과 발열을 줄여준다. Speedstep은 CPU의 전력 제어 중 P-state에 해당하는데 CPU 사용량이 적을 때 코어의 주파수를 낮추어 적은 전압으로 CPU가 동작하도록 설정하게 된다. 본 성능 테스트에서는 최상의 성능을 측정 상황을 구현하기 위해 Speedstep은 사용하지 않고 진행하였다.

Cache & Hardware Prefetcher는 빈도가 높은 데이터의 접근 시간을 줄이기 위해 사용되는 임시 저장소로 주기억장치인 메모리보다 접근시간이 훨씬 빠르다. 보다 빠른 성능 측정을 위해 활성화하여 진행하였다.

Hyper-Threading은 하나의 물리적 유닛에 두 개의 가상 실행 유닛을 할당함으로써 성능을 높이는 기술이다. 운영 체제에서는 코어 한 개당 스레드가 하나씩 추가되어 싱글코어에는 두 개의 코어가, 듀얼코어에는 네 개의 코어가 장착된 것으로 인식하게 된다. 단일 보안 연계에서 성능 측정 시 결과 값에 영향을 미치지 않으며〔2〕, G/W에서 사용한 CPU에서 지원하지 않는 기술이므로 동작여부를 고려하지 않고 진행하였다.

VPN 구현을 위해 사용한 StrongSwan은 패키지 형태(RPM, DEB)인 Currrent Release와 개발 가능한 형태의 Developer Release 두 가지로 나뉘어 제공된다. 〔Table 5〕에서는 StrongSwan에서 기본적으로 제공되는 플러그인으로 AES의 운영 모드 중 CBC 모드만을 포함하고 있다.

Currrent Release로 설치 한 StrongSwan의

경우 [Table 5]에서 제공하는 기본 플러그인 기능만 사용할 수 있으며, [Fig. 15]와 같이 VPN에서 로드된 플러그인을 확인할 수 있다.

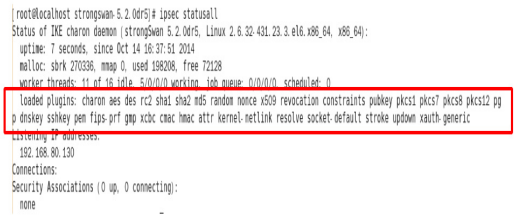


Fig. 15. Loaded Plugins

Table 5. Enabled by default plugin list

plugin	Note
aes	128/192/256 bit key length (Only CBC mode)
cmac	MAC functions
constraints	X.509 certificate
des	DES/3DES encryption
dnskey	DNS Public key
fips-prf	EAP-SIM/AKA algorithm
gmp	RSA/DH code
hmac	MAC function
md5	HASH function
nonce	Random number generation
pem	PEM encoding/decoding
pgp	PGP encoding/decoding
pkcs1	PKCS#1 encoding/decoding
pkcs7	PKCS#7 encoding/decoding
pkcs8	PKCS#8 encoding/decoding
pkcs12	PKCS#12 encoding/decoding
pubkey	RAW Public key process
random	/dev/[u]random RNG process
rc2	RC2 code software
revocation	X.509 CRL/OSCP revocation
sha1	SHA1 hash function
sha2	SHA 256/384/512 hash function
sshkey	SSH key decoding function
x509	CRLs/OSCP message
xcbc	block cipher modes of operation

StrongSwan의 데몬(Charon)이 로드하게 되는 플러그인 디렉토리는 [Fig. 16]과 같이 구성되어있다.

하지만 AES 알고리즘을 사용하는 범용적인 분야에서 적용 가능한 AES-NI 명령어 집합을 본 논문에서 제안하는 IPSec 기반 VPN에 최적화하기 위해서는 AES의 3가지 키 길이뿐만 아니라 각 키 길이의 암호 운영 모드 별 성능 비교가 필요하다. 이를 위해 Developer Release의 libstrongswan을 추가로 빌드하였다. 빌드를 통해 추가된 암호 운영 모드는 [Table 6]과 같다.

Developer Release의 libstrongswan을 추가 빌드 시에 지정한 경로에 [Fig. 17]과 같이 데몬(Charon)이 로드하게 되는 플러그인이 추가로 생성된 것을 확인할 수 있다.

또한 AES의 암호 운영 모드 중 본 논문에서 주장하고자 하는 GCM에 최적화 된 VPN 환경을 구현하기 위해 Intel에서 제안하는 AES-NI의 성능 검증 문서[2]에서 제안하는 Linux AES-NI-GCM Crypto Plug-in Design을 기반으로 [Fig. 16]와 같이 구현하였다. AES-NI의 PCLMULQDQ 명령어의 지원을 받아 GMAC 기반의 GCM 모드를 동작하기 위한 환경을 구성하였다.

AES-NI-GCM을 사용하기 위해서는 [Fig. 18]

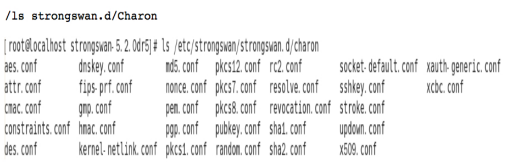


Fig. 16. Charon Plugin list (Deafult Version)

Table 6. Strongswan plugin list

plugin	note
ctr	block cipher modes of operation
ccm	block cipher modes of operation
gcm	block cipher modes of operation

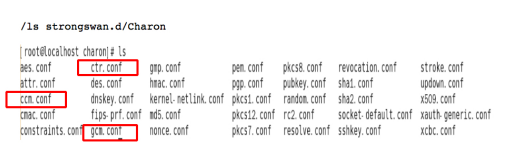


Fig. 17. Charon Plugin list (Build Version)

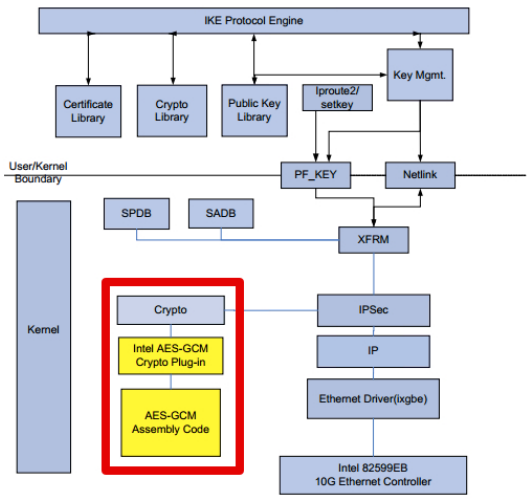


Fig. 18. Linux AES-NI-GCM Crypto Plug-in

의 Kernel Level에서 동작되는 Crypto Driver에 AES 알고리즘의 암·복호화를 위해 Linux Kernel 2.6 이상에서 제공되는 aesni-intel_glue.c 뿐만 아니라 어셈블리 언어 형태의 aesni-intel_asm.s를 필요로 한다.

StrongSwan 플러그인 빌드 후 Intel AES-NI Driver를 정상 구현 하였다면, 〔Fig. 19〕과 같이 insmod로 필요한 오브젝트 모듈을 로드하여, 정책이 명시 된 ipsec.conf 파일로부터 보안 연계를 시도하게 된다.

그리고 〔Fig. 20〕과 같이 정상적으로 빌드 한 플

```
[root@192 ~]# ipsec start
Starting strongSwan 5.2.0dr5 IPsec [starter]...
insmod /lib/modules/2.6.32-431.el6.x86_64/kernel/net/key/af_key.ko
insmod /lib/modules/2.6.32-431.el6.x86_64/kernel/net/ipv4/ah4.ko
insmod /lib/modules/2.6.32-431.el6.x86_64/kernel/net/ipv4/esp4.ko
insmod /lib/modules/2.6.32-431.el6.x86_64/kernel/net/xfrm/xfrm_ipcomp.ko
insmod /lib/modules/2.6.32-431.el6.x86_64/kernel/net/ipv4/ipcomp.ko
insmod /lib/modules/2.6.32-431.el6.x86_64/kernel/net/ipv4/tunnel4.ko
insmod /lib/modules/2.6.32-431.el6.x86_64/kernel/net/ipv4/xfrm4_tunnel.ko
```

Fig. 19. Running StrongSwan

```
/ipsec statusall

[root@localhost charon]# ipsec statusall
Status of IKE charon daemon (strongSwan 5.2.0dr5, Linux 2.6.32-431.23.3.el6.x86_64, x86_64):
  uptime: 51 minutes, since Oct 14 15:36:32 2014
  malloc: sbrk 270336, mmap 0, used 226800, free 43536
  worker threads: 11 of 16 idle, 5/0/0/0 working, job queue: 0/0/0/0, scheduled: 0
  loaded plugins: charon aes ccm ctr gcm des rc2 sha1 sha2 md5 random nonce x509 revocation constraints pubkey pkcs1 pkcs7 pkcs8 pkcs12 pgp dnskey sshkey pem fips-prf pkp xcbc cmac hmac attr kernel-netlink resolve socket-default stroke updown xauth-generic
Listening IP addresses:
  192.168.80.130
Connections:
      net-net:  192.168.80.130...192.168.80.131  IKEv2
      net-net:   local:  [10.1.1.10] uses public key authentication
      net-net:   cert:  "C=KO, ST=SS, L=LL, O=HUSL, CN=10.1.1.10, E=10.1.1.10@strong.org"
      net-net:   remote: [10.2.2.10] uses public key authentication
      net-net:   child:  dynamic === dynamic TUNNEL
Security Associations (0 up, 0 connecting):
  none
```

Fig. 20. VPN Status

러그인이 로드된 것을 확인 할 수 있다.

StrongSwan의 G/W간의 인증 방식으로 PSK를 이용하였다. 최적화된 성능 측정을 위한 폐쇄 네트워크에서 CA ROOT 인증서를 자체 발급하였으며, 각 G/W의 인증서를 CA 인증서로 서명하였다. 자체 서명 인증서를 사용하여 보안 연계가 수립되어 〔Fig. 21〕과 같이G/W간의 IPSec 암호화 통신이 연결됨을 확인하였다.

보안 연계가 수립된 후 StrongSwan이 빌드 된 경로에 위치한 "ipsec.conf"를 통해 각 G/W에 연결하게 될 Host PC의 Subnet을 정의할 수 있다. G/W로부터 Subnet 네트워크 대역을 이용하는 Host는 G/W 간의 통신에서 IPSec 기반의 암호화 데이터를 주고받게 된다. G/W에 Host를 동반하여 정상적으로 보안 연계가 수립됨을 확인하였다. 이때 Host는 ESP 패킷(Tunnel Mode로 동작)을 주고받도록 설정하였다〔Fig. 22〕.

G/W에 tcpdump를 이용하여 연결된 Host간의 ping 데이터가 ESP로 암호화 됨〔Fig. 23〕을 확인하였다.

〔Fig. 24〕와 같이 IPSec에서 성능 측정을 위해 사용되는 IKE와 ESP의 알고리즘〔9〕 및 해쉬 함수,

```
07[IKE] initiating IKE_SA net-net[1] to 192.168.1.10
07[ENC] generating IKE_SA_INIT request 0 [ SA KE No N(NATD_S_IP) N(NATD_D_IP) ]
07[NET] sending packet: from 192.168.2.10[500] to 192.168.1.10[500] (992 bytes)
09[NET] received packet: from 192.168.1.10[500] to 192.168.2.10[500] (393 bytes)
09[ENC] parsed IKE_SA_INIT response 0 [ SA KE No N(NATD_S_IP) N(NATD_D_IP) CERTREQ N(MULT_AUTH) ]
09[IKE] received cert request for "C=KO, ST=SS, O=HUSL, CN=ca@strong, E=husl@strong.org"
09[IKE] sending cert request for "C=KO, ST=SS, O=HUSL, CN=ca@strong, E=husl@strong.org"
09[IKE] authentication of '192.168.2.10' (myself) with RSA signature successful
09[IKE] sending end entity cert "C=KO, ST=SS, L=LL, O=HUSL, CN=192.168.2.10, E=host@strong.org"
09[IKE] establishing CHILD_SA net-net
09[ENC] generating IKE_AUTH request 1 [ IDi CERT N(INIT_CONTACT) CERTREQ IDr AUTH SA TSi TSr N(MULT_AUTH) N(EAP_ONLY) ]
09[NET] sending packet: from 192.168.2.10[500] to 192.168.1.10[500] (1570 bytes)
10[NET] received packet: from 192.168.1.10[500] to 192.168.2.10[500] (1421 bytes)
10[ENC] parsed IKE_AUTH response 1 [ IDr CERT AUTH SA TSi TSr N(AUTH_LFT) ]
10[IKE] received end entity cert "C=KO, ST=SS, L=LL, O=HUSL, CN=192.168.1.10, E=host@strong.org"
10[CFG]   using certificate "C=KO, ST=SS, L=LL, O=HUSL, CN=192.168.1.10, E=host@strong.org"
10[CFG]   using trusted ca certificate "C=KO, ST=SS, O=HUSL, CN=ca@strong, E=husl@strong.org"
10[CFG] checking certificate status of "C=KO, ST=SS, L=LL, O=HUSL, CN=192.168.1.10, E=host@strong.org"
```

Fig. 21. Security Association

```
[root@192 ~]# ipsec statusall
Status of IKE charon daemon (strongSwan 5.2.0dr5, Linux 2.6.32-431.el6.x86_64, x86_64):
  uptime: 13 seconds, since Jun 18 20:36:06 2014
  malloc: sbrk 270336, mmap 0, used 227808, free 42528
  worker threads: 11 of 16 idle, 5/0/0/0 working, job queue: 0/0/0/0, scheduled: 3
  loaded plugins: charon aes des rc2 sha1 sha2 md5 random nonce x509 revocation constraints pubkey pkcs1 pkcs7 pkcs8 pkcs12 pgp dnskey sshkey pem fips-prf gmp xcbc cmac hmac attr kernel-netlink resolve socket-default stroke updown xauth-generic
Listening IP addresses:
  192.168.2.10
  10.2.2.1
Connections:
      net-net:  192.168.2.10...192.168.1.10  IKEv2
      net-net:   local:  [192.168.2.10] uses public key authentication
      net-net:   cert:  "C=KO, ST=SS, L=LL, O=HUSL, CN=192.168.2.10, E=host@strong.org"
      net-net:   remote: [192.168.1.10] uses public key authentication
      net-net:   child:  10.2.0.0/16 === 10.1.0.0/16 TUNNEL
Security Associations (1 up, 0 connecting):
      net-net[1]: ESTABLISHED 13 seconds ago, 192.168.2.10[192.168.2.10]...192.168.1.10[192.168.1.10]
      net-net[1]: IKEv2 SPIs: f2158b05e4d7bc03_i* 228d8f32b5063754_r, public key reauthentication in 52 minutes
      net-net[1]: IKE proposal: AES_GCM_12_128/PRF_HMAC_SHA1/MODP_1536
      net-net[1]:  INSTALLED, TUNNEL, ESP SPIs: cfcc7727_i cb2693ae_o
      net-net[1]:  AES_GCM_12_128, 0 bytes_i, 0 bytes_o, rekeying in 15 minutes
      net-net[1]:   10.2.0.0/16 === 10.1.0.0/16
```

Fig. 22. Host Subnet

```
[root@192 crypto]# tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
20:37:19.350778 IP 192.168.2.10 > 192.168.1.10: ESP(spi=0xcb2693ae,seq=0x10), length 116
20:37:19.351196 IP 192.168.1.10 > 192.168.2.10: ESP(spi=0xcfcc7727,seq=0x10), length 116
20:37:19.351196 IP 10.1.1.1 > 10.2.2.10: ICMP echo reply, id 15660, seq 15, length 64
20:37:20.350768 IP 192.168.2.10 > 192.168.1.10: ESP(spi=0xcb2693ae,seq=0x11), length 116
20:37:20.351227 IP 192.168.1.10 > 192.168.2.10: ESP(spi=0xcfcc7727,seq=0x11), length 116
20:37:20.351227 IP 10.1.1.1 > 10.2.2.10: ICMP echo reply, id 15660, seq 17, length 64
20:37:21.350743 IP 192.168.2.10 > 192.168.1.10: ESP(spi=0xcb2693ae,seq=0x12), length 116
```

Fig. 23. Ping-ESP Check

```
conn %default
                ike=aes128gcm12-sha1-modp1536
                esp=aes128gcm12
                ikelifetime=60m
                keylife=20m
                rekeymargin=3m
                keyingtries=1
                keyexchange=ikev2
                mobike=no
conn net-net
                left=192.168.2.10
                leftsubnet=10.2.0.0/16
                rightsubnet=10.1.0.0/16
                leftcert=newCert.pem
                leftid=192.168.2.10
                leftprotoport=%any
                leftfirewall=yes
                right=192.168.1.10
                rightid=192.168.1.10
                rightprotoport=%any
                auto=start
```

Fig. 24. ipsec.conf

키 갱신 주기, 보안 연계를 수립할 IP 정보 및 서브넷 정보 등을 "ipsec.conf" 파일에서 정의할 수 있다.

3.2 성능 측정

본 논문에서 성능 측정을 위해 비교하는 알고리즘은 AES의 3가지 키 길이(128, 192, 256)와 각 키 길이 별 4가지 운영 모드(CBC, CTR, CCM, GCM)이다. 성능 측정 비교를 위해 각 G/W 간의 IPSec이 동작 유무와 관계없이 1Gbps 회선에서 1024 패킷 사이즈의 성능이 968Mbps로 동일함을 확인한 후 (IPSec이 동작되지 않을 시 AES-NI에 의한 성능의 영향을 받지 않음을 확인) 진행하였다. 3DES-SHA1 알고리즘에서도 AES-NI에 의한 성능 차이가 없음을 확인하였다.

성능 측정은 KISA에서 보유 중인 IXIA 장비를 이용하여, 〔Fig. 14〕와 같이 네트워크를 구성하였다. IXIA 장비에서 제공하는 Automate 툴을 사용하여, 각 패킷 사이즈(64, 128, 256, 512, 1024) 별 송수신 패킷 사이의 손실률이 0인 값을 측정하였다. 일반적으로 VPN은 본사-지사 관계의 형태로 다중 보안 연계를 수립하여 이용되지만, 연구를 위해 사용되는 PC의 NIC의 한계로 단일 보안 연계를 수립하여 측정하였다. 일반적으로 VPN은 〔Fig. 25〕와 같이 본사-지사 관계의 형태의 다중 보안 연계를 수립하여 이용된다. 본사의 네트워크 서버가 물리적으로 거리가 먼 경우 보안에 취약한 상용 회선을 이용하되 지사와 본사의 안전한 통신을 위해 보안 연계를 수립하게 되는데, 지사의 수가 하나 이상 일 경우

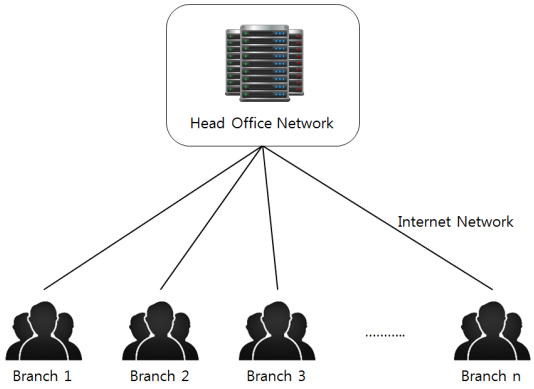


Fig. 25. Multiple Security Associations

다중 보안 연계가 수립되게 된다.

본 논문에서는 국내 보안 업체에서 제품의 성능을 테스트하기 위해 실제로 사용하는 방식과 동일한 환경인 계측기를 이용한 단일 보안 연계 상에서의 성능 측정만을 측정하였다. 단일 보안 연계 구성이 실제 본사-지사 관계를 표현하면 〔Fig. 26〕과 같다.

패키지 형태로 설치 한 StrongSwan에서 제공되는 기본 플러그인〔Table 5〕으로 AES-NI 명령어 집합을 이용해 본 결과 AES-NI의 지원을 받더라도 1024사이즈 패킷 기준 약 192Mbps 전후의 성능 차이를 보였다〔Table 7〕.

〔Fig. 27〕에서 보는 것과 같이 IPSec을 OFF 했을 경우와 3DES-CBC 알고리즘을 사용한 경우 AES-NI의 지원 유무에 의한 성능 차이가 발생하지 않았고, AES-CBC 모드에서는 평균 약18%의 성능 향상을 보였다.

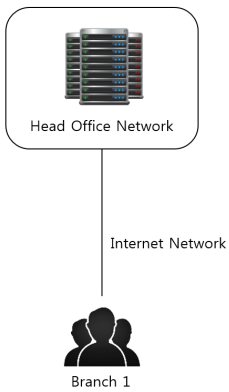


Fig. 26. Single Security Association

Table 7. Default Plugin (Speed: Mbps)

algorithm/AES-NI	ON	OFF
IPSec OFF	978	980
3DES CBC-SHA1	177	178
AES128 CBC-SHA1	712	518
AES192 CBC-SHA1	707	514
AES256 CBC-SHA1	694	504

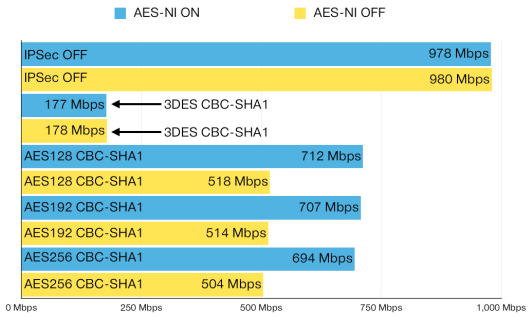


Fig. 27. Default Plugin

본 논문에서는 AES-NI의 최적화 된 G/W 환경을 구현하기 위해 Strongswan을 Developer Release로 빌드하여 플러그인 형태의 AES의 4가지 암호 운영 모드를 추가하여 성능을 측정하였으며, AES-NI-GCM Crypto Plug-in Design을 기반으로 PCLMULQDQ 명령어의 지원을 받아 GMAC 기반의 GCM 모드를 동작하기 위한 환경을 구성하였다. 측정한 IPSec의 성능은 AES-NI의 동작 유무에 따라 〔Table 8, Table 9〕과 같다.

Table 8. AES-NI ON (Speed: Mbps)

algorithm/ packet	64	128	256	512	1024
IPSec OFF	95	226	416	674	968
3DES CBC-SHA1	89	99	136	162	176
AES128 CBC-SHA1	108	107	264	449	732
AES192 CBC-SHA1	68	115	263	445	736
AES256 CBC-SHA1	95	108	290	361	732
AES128 CTR-SHA1	98	162	283	484	722
AES192 CTR-SHA1	105	155	275	486	773
AES256 CTR-SHA1	112	221	318	658	889
AES128 CCM12	104	162	263	239	595
AES192 CCM12	37	166	275	422	589
AES256 CCM12	41	80	273	422	554
AES128 GCM12	89	90	396	481	935
AES192 GCM12	95	170	346	634	934
AES256 GCM12	106	170	347	634	934

Table 9. AES-NI OFF (Speed: Mbps)

algorithm/ packet	64	128	256	512	1024
IPSec OFF	113	225	412	672	968
3DES CBC-SHA1	87	125	152	169	180
AES128 CBC-SHA1	105	147	245	407	568
AES192 CBC-SHA1	100	144	242	392	542
AES256 CBC-SHA1	101	141	242	384	524
AES128 CTR-SHA1	102	147	246	406	697
AES192 CTR-SHA1	105	145	242	391	537
AES256 CTR-SHA1	105	139	234	387	633
AES128 CCM12	107	169	264	423	582
AES192 CCM12	107	158	249	395	531
AES256 CCM12	108	151	243	373	481
AES128 GCM12	36	218	409	542	890
AES192 GCM12	107	153	273	460	713
AES256 GCM12	40	54	237	361	590

IV. 측정 결과 분석

측정 결과의 분석은 1024 사이즈 패킷을 기준으로 각 키 길이 별 운영모드의 대역폭을 비교하였다. 모든 성능 측정은 단일 보안 연계 수립으로 측정 하였다.

AES128에서는 AES-NI의 지원을 받을 경우와 지원을 받지 않을 경우 4가지 운영모드의 평균 약 10%의 성능 향상을 보였다. 각 운영모드 별로는 CBC 모드에서 약 29%의 가장 큰 성능 향상을 보였으며, 인텔이 AES-NI의 성능 향상을 위해 제안하는 운영 모드인 GCM[2,3]에서는 약 5%의 성능 차이를 보였다. 또한 CTR 모드에서 약 4%, CCM에서 약 2%의 성능 향상을 보였다.

AES192에서는 AES-NI의 지원을 받을 경우와 지원을 받지 않을 경우 4가지 운영모드의 평균 약 30%의 성능 향상을 보였다. 각 운영모드 별로는 CTR 모드에서 약 44%의 가장 큰 성능 향상을 보였으며, 인텔이 AES-NI의 성능 향상을 위해 제안하는 운영 모드인 GCM[2,3]에서는 약 31%의 성능 차이를 보였다. 또한 CBC 모드에서 약 36%, CCM에서 약 11%의 성능 향상을 보였다.

AES-256에서는 AES-NI의 지원을 받을 경우와 지원을 받지 않을 경우 4가지 운영모드의 평균 약 38%의 성능 향상을 보였다. 각 운영모드 별로는 인텔이 AES-NI의 성능 향상을 위해 제안하는 운영 모드인 GCM[2,3]에서 약 58%의 가장 큰 성능 향상을 보였다. 또한 CBC 모드에서 약 40%, CTR 모드에서 약 40%, CCM에서 약 15%의 성능 향상을 보였다.

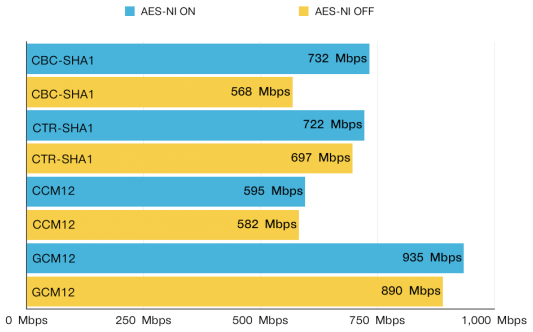


Fig. 28. AES128 - Packet 1024

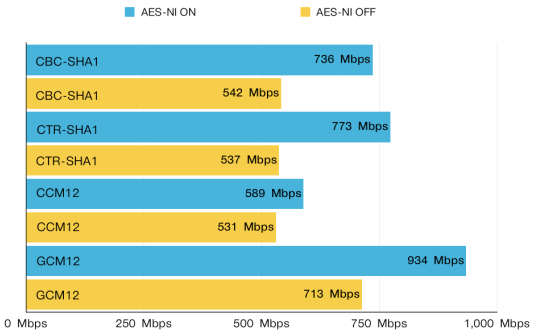


Fig. 29. AES192 - Packet 1024

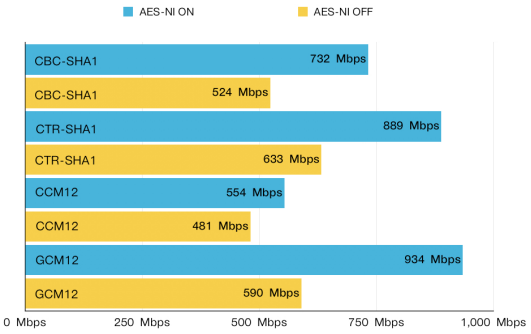


Fig. 30. AES256 - Packet 1024

IXIA 장비를 이용하여 측정한 결과 AES-NI의 지원을 받은 모든 ESP(Packet-1024) 패킷의 성능이 향상되었다[Table 10].

Table 10. Packet-1024 (Speed: Mbps)

algorithm	AES-NI	None	performance	Improvement
AES128 CBC-SHA1	732	568	164	29%
AES192 CBC-SHA1	736	542	194	36%
AES256 CBC-SHA1	732	524	208	40%
AES128 CTR-SHA1	722	697	25	4%
AES192 CTR-SHA1	773	537	236	44%
AES256 CTR-SHA1	889	633	256	40%
AES128 CCM12	595	582	13	2%
AES192 CCM12	589	531	59	11%
AES256 CCM12	554	481	72	15%
AES128 GCM12	935	890	45	5%
AES192 GCM12	934	713	221	31%
AES256 GCM12	934	590	344	58%

V. 결 론

AES 알고리즘을 사용하는 범용적인 분야에 적용 가능한 인텔 칩의 AES-NI 명령어 집합은 Linux Kernel에서 점차 그 범위를 확대[4]하고 있으며, BIOS에서 AES-NI 지원을 ON/OFF 하는 것만으로는 PCLMULQDQ 명령어의 지원을 받지 못하여 최적의 성능을 사용하기 위해 별도의 환경 구현이 필요하다.

본 논문에서는 인텔에서 권장하는 AES-NI 명령어 집합을 IPSec 기반 VPN에서 오픈 소스 빌드 후 CBC 모드만을 제공하기 때문에 최적화하기 위해 플러그인 디자인을 구현하여 AES의 CBC 암호 운영 모드뿐만 아니라 CTR, CCM, GCM의 각 키 길이 별 계측을 통해 IPSec 기반 VPN의 성능을 최적화 하였다. 객관적인 성능 측정을 위해 국내 보안 업체에서 장비의 성능을 검증하기 위해 사용하는 IXIA 로 ESP 패킷의 암·복호화를 통한 가용 회선 대역폭 성능을 측정하였다. 단일 보안 연계 수립 시 AES의 키 길이와 상관없이 가장 우수한 성능을 보인 알고리즘은 인텔에서 권장하는 AES의 운영 모드인 GCM을 이용한 AES128 GCM(935Mbps)로 측정 되었다. 각 키 길이별 성능 측정에서도 모두 AES GCM의 성능이 가장 우수한 것으로 측정되었다[Table 8]. AES-NI 유무에 따른 성능 향상은 키 길이와 상관없이 AES256 GCM에서 약 58%로 가장 높은 성능 향상을 보였다. 또한 각 키 길이별 성능 향상은 128bit 일 때 CBC 모드에서 약 29%, 192bit 일 때 CTR 모드에서 약 44%, 256bit 일 때 GCM에서 약 58% 향상 되었다.

따라서 본 논문에서는 Intel 칩의 AES-NI를 이용한 IPSec 기반 VPN의 AES 알고리즘의 단일 보안 연계 시 AES128 GCM을 사용하는 것이 가장 좋다는 것을 알 수 있었다.

다중 보안 연계가 이루어지는 VPN 환경에서 사용할 경우에는 키 길이를 조절해가며 사용할 것을 권장한다. 다만 비교 대상인 제조사 VPN의 OS 최적화, AES의 암호 운영 모드 및 키 길이 등 공식적인 자료가 없어 절대적인 비교는 힘들다.

본 연구를 기반으로 차기 논문에서는 AES-NI의 7가지 명령어 집합을 이용하여 국내에서 이용이 권장되고 있는 블록 알고리즘인 ARIA, SEED의 성능을 개선하고자 한다.

References

[1] Kahraman Akdemir, "Breakthrough AES Performance with Intel AES New Instructions," White Paper, 2010.

[2] Adrian Hoban, "Using Intel AES-New Instructions and PCLMULQDQ to Significantly Improve IPSec Performance on Linux," White Paper, 324238-001, Aug. 2010.

[3] Vinodh Gopal, "Optimized Galois-Counter-Mode Implementation on Intel Architecture Processors," Aug. 2010

[4] "Intel Advanced Encryption Standard New Instructions(AES-NI) Ecosystem March 2013 Update http://www.intel.com/content/dam/www/public/us/en/documents/specification-updates/aes-ni-ecosystem-update.pdf

[5] Oracle Solaris, x86 Intel AES-NI Optimization, http://docs.oracle.com/cd/E19253-01/821-2301/gjxnz/

[6] Hyunjin Ahn and Choi Wanseong, "Performance comparison of ARIA-CCM and ARIA-GCM in ARM9", "Kookmin University, KISA," "KICS" 108-109

[7] strong swan User Documentation Ipsec.conf, http://wiki.strongswan.org/projects/strongswan/wiki/IKEv2CipherSuites

〈저자소개〉

정 진 표 (Jin-Pyo Jeong) 정회원
2005년 2월: 한세대학교 정보통신공학과 졸업
2008년 8월~2012년 8월: 성균관대학교 정보보안학과 석사과정 수료
2006년 4월~2010년 1월: ㈜시큐아이 기술지원팀 근무
2010년 1월~2011년 2월: ㈜이니텍 보안솔루션팀 근무
2011년 7월~2012년 8월: ㈜이글루시큐리티 기술지원센터 근무
2013년 11월 ~ 현재: ㈜아이티센 공공사업부
〈관심분야〉 네트워크 보안, 리눅스 시스템, 개인정보보호 등

황 준 호 (Jun-Ho Hwang) 학생회원
2013년 2월: 한세대학교 정보통신공학과 졸업
2013년 3월~현재: 한세대학교 IT융합학과 석사과정
〈관심분야〉 리눅스 시스템, 시스템 취약점, 모바일 보안 등

한 근 희 (Keun-Hee Han) 종신회원
서울과학기술대학교 컴퓨터공학과 졸업
한양대학교 과학대학원 공학석사
고려대학교 대학원 이학박사
현재: 고려대학교 융합소프트웨어전문대학원 산학교수
〈관심분야〉 소프트웨어 보증, 시큐어 코딩, 정보보호관리 체계, 개인정보보호, 클라우드 컴퓨팅 보안, 스마트 의료 보안, 스마트 자동차 보안 등

김 석 우 (Seok-Woo Kim) 종신회원
1979년: 한국항공대학교 통신공학과과 졸업(학사)
1989년: 미국 NJIT 전자계산학과(공학석사)
1995년: 아주대학교 일반대학원 컴퓨터 공학과(공학박사)
1980년~1997년: 한국전자통신연구원 책임연구원, 부호5실장
2000년~현재: 한세대학교 정보통신학과 교수
〈관심분야〉 시스템 보안, 네트워크 보안, 시스템 평가 등