IPv6용 IPSec 하드웨어 칩을 위한 AES 모듈

김경태° 김지욱* 박상길° 양인제° 김동규° 이정태°
°부산대학교 컴퓨터공학과
*부산대학교 정보시스템공학과

AES Modules for IPSec Hardware Chips in IPv6

Kyung-Tae Kim Ji-Wook Kim Sang Kil Park Inje Yang Dong Kyue Kim Jungtae Lee
Department of Computer Engineering, Pusan National University
Department of Information System Engineering, Pusan National University
{ktkim, jwkim8, skpark, injepost, dkkim1, jtlee}@pusan.ac.kr

요 약

급속히 고갈되어가는 IPv4의 주소 부족 문제를 해결하기 위하여, 차세대 인터넷 프로토콜(IP)인 IPv6가 제안되었고 실용화 단계까지 진행되고 있다. IPv6에서의 요구 사항 중의 하나인 IPSec은 IPv4의 취약한 보안 기능을 강화하는 것이다. 현재 IPSec에서 반드시 구현되어야 할 암호화 알고리즘으로 MD5, SHA1, 3DES 와 더불어 최근 표준안으로 채택된 AES(Rijndael)을 요구하고 있다. IPv6의 고속 수행을 위하여는 IPSec이 하드웨어로 구현될 필요성이 있으므로, 본 논문에서는 IPv6용 IPSec 칩에 탑재할 AES 하드웨어 모듈을 구현하였다. 제안된 하드웨어 모듈은 효율적인 알고리즘의 수행과 구현을 위하여, 암호화/복호화 단계가 동일한 구조로 동작하도록 설계하였으며, 가변적인 128, 196, 256 비트의 키에 대하여 같은 로직을 사용하도록 설계하였다.

1. 서론

최근 언제 어디서나 원하기만 하면 인터넷에 접속할 수 있는 환경을 구축하려는 요구와 노력이 점차 확대되고 있다. 이러한 추세에 따라, PDA와 핸드핼드 컴퓨터 등의 이동 단말기 기술들이 발전하고 있으며, 모바일 통신 서비스에 대한 수요는 폭발적으로 증가하고 있는 실정이다.

그러나 이와 같은 미래의 Post-PC 시대에 기존의 인터넷 프로토콜인 IPv4를 사용해 수많은 인터넷 단말들을 모두 수용하고 다양한 서비스를 제공하기에는 주소 할당에 한계가 있다. 이런 문제를 해결하기 위하여, 차세대 인터넷 프로토콜(IP)인 IPv6가 제안되었고[1] 실용화 단계까지 진행되고 있다.

IPv6에서는 IPv4에서는 하나의 선택 사항이었던 IPSec을 필수적인 사항으로 요구한다[2]. IPSec은 IPv4의 취약한 보안 기능을 강화하는 것으로써, IP 계층에서 데이터 암호화를 제공하여 IP 패킷의 자체적인 보안 수단을 제공하는 것이다.

현재 IPSec에서 반드시 구현되어야 할 암호화 알고리즘으로 MD5, SHA1, 3DES 와 더불어 최근 표준안으로 채택된 AES(Rijndael)[3]을 요구하고 있다. 이미 표준으로 채택된 알고리즘들은 소프트웨어나 하드웨어로 많이 구현되어 있는 실정이다. 그러나 IPSec에

적용된 알고리즘들은 모두 IPv4용으로 운영체제(Operating System)내의 IPv4 스택과 연동하여 동작한다. 본 연구에서는 IPv6와 연동하여 인터페이스하는 IPSec을 하드웨어로 구현하는 데에 그 목적이 있다

이를 위하여, 본 논문에서는 IPSec에서 요구하는 암호화 알고리즘들 중 가장 최근 표준으로 채택된 AES의 하드웨어 모듈을 IPv6용으로 설계 및 구현하였다. 제안된 하드웨어 모듈은 효율적인 알고리즘의 수행과 구현을 위하여 다음과 같은 장점을 가지도록 설계하였다.

1) AES의 구현시 단점을 보완하여 암호화/복호화 단계가 동일한 구조로 동작한다.
2) 가변적인 128, 196, 256 비트의 키에 대하여 동일한 로직을 사용한다.

본 논문의 구성은 다음과 같다. 먼저, 2장에서 IPv6와 AES의 기본지식을 기술하고, 3장에서는 IPv6용 AES하드웨어 모듈의 설계 방법을 설명한 후, 4장에서 성능분석과 결론을 맺는다.

2. 기본 지식

2.1 IPv6와 IPSec

기하급수적으로 증가하는 다양한 인터넷 단말들에게 고유한 주소를 부여하고자 한다면, 기존 IPv4의 32 비트 주소 체계로는 이를 모두 수용할 수 없다. 따라서, 이 같은 주소 부족 문제를 해결하기 위한 방안으로 IETF에 의해 128비트로 주소를 확장한IPv6가 제안되었다. 차세대 IP인 IPv6는 IPv4와 달리 실시간 서비스 지원, 보안 지원, 자동 설정 기능(auto-configration), 이동 호스트의 지원을 포함한 강화된 라우팅 기능을 제공한다[1].

IPv6에서 보안 기능의 강화는 IPSec (IP Security)을 구현하므로써 이루어진다. IPSec은 IPv4를 설계 할 당시, 보안을 고려하지 않아 발생되는 IP Spoofing, IP sniffing과 같은 보안 허점을 IP 계층 자체에서 해결하기 위한 방안이다.

IPv6에서는 IPSec의 AH(Authentication Header) 프로토콜[4]과 ESP(Encapsulating Security Payload) 프로토콜[5]을 위한 패킷 헤더를 정의하고 구현한다. AH에는 암호화 알고리즘인 MD5, SHA1등의 동작으로 생성된 메시지 인증 코드가 첨부되어 IP 패킷의 무결성과 인증을 제공한다. 또한, ESP에서는 전송하고자 하는 평문 데이터를 암호화한 데이터가 암호화 알고리즘인 3DES, AES 등의 동작으로 생성된 후, 첨부되어 IP 데이터 그램의 기밀성을 제공한다.

2.2 AES

2000년말에 표준안으로 선택된 AES(Rijndael)은 대칭키 구조를 가지는 암호화/복호화 알고리즘으로, DES[6]의 64비트의 키 크키를 128비트 이상의 키로 확장한 알고리즘이다. AES의 알려진 특징을 정리하면 다음과 같다.

- 기존의 공격 (linear 및differential 공격 등)에 안전하도록 S-box를 설계하였으며, 바이트 단위로 병렬적으로 처리될 수 있다.
- 각 라운드(round)의 변환이 Feistel 구조를 가지지 않으며, 여러 라운드 수행이 높은 확산 효과를 보장한다.
- 역변환이 가능하도록 uniform transformation을 지원한다.

키의 크기는 128, 196, 256 비트를 가질 수 있는 데, 이 키의 크기(Nk)는 [그림 1]에서와 같이 데이터 블럭(data block)을 암호화/복호화하는 라운드 수(Nr)와 확장될 키의 전체 길이를 결정한다. 표준에서 데이터 블록의 크기(Nb)는 128비트 (즉, Nb =4)로 고정되어 있다

키의 크기	Nb	Nr	Nk	W(Nk*(Nr+1)
128 bit	4	10	4	44
196 bit	4	12	6	52
256 bit	4	14	8	60

Nb : Block의 크기(32bit 워드 단위)
Nr : 라운드의 횟수
Nk: 키의 크기의 크기(32bit 워드 단위)
W : 확장된 키의 전체길이(32bit의 워드 배열)

[그림 1] 키의 크기별 라운드와 확장된 키의 크기

라운드 키 K^r은 $r \in \{0,\ldots,Nr\}$일 때, 확장된 키의 배열인 W[4r],...,W[4r+3]로 이루어진다. 이를 이용하여

Nr 라운드를 수행하는AES알고리즘은 [그림 2]와 같은 구조를 가지고 반복 수행된다. 각 라운드에서는 BS(S-Box), SR(Shift Row), MC(Mix Column)을 수행한 뒤 확장된 키와 XOR연산을 수행하는 KA를 수행한다[3].

1. S := plaintext;
2. KA (S, K); (Add Round Key)
3. *for r:=1 to R do* (R번 라운드 수행)
   A. S:=BS(S); (Byte Substitution)
   B. S:=SR(S); (Shift Row)
   C. S:=MC(S); (Mix Column)
   D. S=KA(S,K); (Add Round Key)
4. ciphertext := S;

[그림 2] AES 알고리즘

3. IPv6용 AES FPGA 보드의 설계

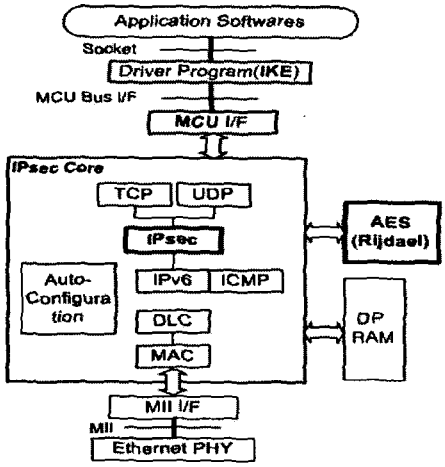
3.1 IPv6용 IPSec의 전체 모듈 설계

IPv6용 AES FPGA 보드는 **AES** 모듈과 **MCU, IKE, IPSec** 코어(Core) 모듈의 네 부분으로 구성되며 [그림 3]에 블록 다이어그램으로 나타내었다. AES모듈은 3.2절에서 자세히 기술하며, 나머지 세 모듈은 다음과 같다.

MCU는 드라이버 프로그램을 읽어 들여 수행하며, 읽어 들인 드라이버 프로그램의 내용에 따라 전체 모듈을 제어하는 역할을 한다.

IKE는 상대편 IPv6호스트와 통신하기 앞서 사용할 암호화 알고리즘과 암호화 키를 협상하여 교환하는 역할을 한다. IKE는 MCU에 의해 수행될 드라이버 프로그램의 일부분으로 구현되고, IKE의 수행 결과로 얻어진 암호화 알고리즘 정보와 암호화 키 정보는 IPSec 코어로 전달되어 IP 패킷을 암호화/복호화할 때 사용된다.

IPSec 코어는 상대편과 통신하기 위한 TCP/IPv6 기본 모듈과 IPSec을 위해 AH, ESP 프로토콜 처리를 위해서 헤더 기능의 추가로 구현된 것이다. 이때 암호화/복호화에 사용될 알고리즘 선택과 암호화 키는 IKE로부터 가져오고, AH와 ESP헤더에 들어갈 암호화 데이터들은 AES칩과 통신을 통하여 생성한다. IPSec 코어에 들어가는 TCP/IPv6 기본 모듈은 연구실에서 기 구현된 것을 사용한다[7]. AES 모듈은 IPSec 코어 모듈의 요청에 따라 암호화할 128비트씩의 평문 데이터와 가변적인128, 192, 256비트의 암호화 키를 인터페이스를 통해 받아들여 구동되며 결과값인 128bit암호화된 데이터를 다시 IPSec 코어 모듈에 넘겨주는 역할을 한다.



[그림 3] IPv6 용 IPSec 의 전체 모듈

3.2 AES 모듈 설계

AES전체 모듈은 [그림 5]에서와 같이 Interface 모듈, KeySchedule 모듈, Controller 모듈, ALG 모듈의 4가지 모듈로 구성된다. 각 모듈의 기능을 간략히 기술하면 다음과 같다[8, 9].

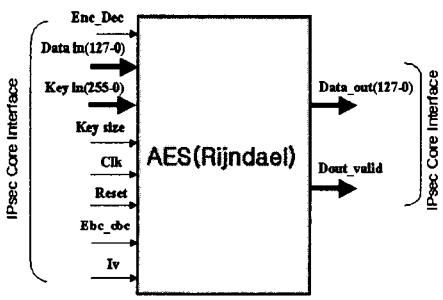
1) **Interface** 모듈 : IPSec 코어 모듈과 통신하기 위한 모 듈로써 데이터, 키, 키의 크기, 초기화 벡터, 동작모 드 선택 신호를 받아들인다.
2) **KeySchedule** 모듈: 초기 키가 주어졌을 때 키를 확장한다.
3) **Controller** 모듈 : KeySchedule 모듈과 ALG 모듈의 시작과 정지를 제어한다. 또한, ALG 모듈의 동작모드인 ECB(Electronic Code Book) 모드와 CBC(Cipher Block Chaining)모드 제어한다[10].
4) **ALG** 모듈 : 각 라운드의 확장된 키를 이용하여 각각의 라운드를 수행하는데 Enc_dec의 값에 따라서 암호화/복호화를 결정한다. 이 모듈에서는 FPGA 탑재시 효율적으로 공간을 사용하기 위하여,

암호화와 복호화 과정을 동일한 구조와 데이터 흐름을 가지도록 설계하였다. 또한 Controller에서 받은 동작 모드 신호를 이용해서 기본적으로 CBC 모드로 동작하도록 설계하였다.

3.2.1 인터페이스 모듈 설계

AES 모듈은 IPSec 코어 모듈과 포트맵(port map)으로 연결된다. [그림 4]에서는 두 모듈간의 인터페이스 신호들을 보여주고 있다.

[그림 4]에서 보인 인터페이스 신호들을 기술하면 다음과 같다. 입력 신호중 Enc_dec은 AES 모듈이 암호화 일 때 이진수 값 '1', 복호화 일 때 '0'을 가진다. Data_in은 암호화할 실제 데이터를 128비트씩 입력받고, Key_in은 2비트의 Key_type을 설정하고 키를 입력받는다. 즉, Key_type이 '00'일 때128, '01' 때196, '10' 일 때 256 비트 길이의 키를 입력 받는다. Din_valid와 Dout_valid는 각각 데이터 값 입,출력이 유효하면 '1'로 설정된다. 출력 신호인 Data_out은 AES 모듈의 동작에 의한 128bit의 결과값을 IPSec 코어로 되돌려 보낼 경우 사용되는 신호를 의미한다.



[그림 4] AES 모듈의 인터페이스

3.2.2 KeySchedule 모듈과 컨트롤러 모듈

KeySchedule 모듈에서는 초기 키가 주어졌을 때 키를 확장한다. 즉 확장된 키배열 W를 생성하는데, 각 $W[i] \in \{W[0],..W[Nb*(Nr+1)]\}$의 인덱스 i는 Nk의 값에 따라 결정 된다.

[그림5]에서 보인 KeySchedule 모듈의 내부에서는 모든 확장된키를 미리 생성하여 저장하는 메모리인 Expanded Key 블럭을 사용하여 구현하였다. 실 응용에서는 하나의 키에 대하여 암호화/복호화를 수행할 데이터 블럭이 충분히 많을 수 있으므로, Expanded Key 블럭을 사용하면 공간은 많이 차지 하더라도 속도를 향상시킬 수 있다.

KeySchedule 모듈에서 키의 확장이 종료되면 ALG 모듈로 확장된 키를 넘기게 되고 확장키를 받은 ALG 모듈은 이를 이용해 암호화/복호화를 수행한다.

Controller 모듈은 KeySchedule 모듈과 ALG 모듈의 시작과 정지를 제어한다. 또한 Ecb_cbc신호는 '1'일 때 ALG모듈이 EBC모드로, '0'일 때는 CBC모드로 동작하도록 제어한다.

3.2.3 ALG 모듈 설계

ALG모듈은 [그림 2]에서 각 라운드에서 필요한 BS(S-Box), SR(Shift Row), MC(Mix Column), KA부분 단계를 4가지 부분 모듈로 구성하였다. 각 부분 모듈에 대하여 다음과 같이 설계하였다.

①**ByteSub/InvByteSub**는 암호화시 S-box에 의하여 각 바이트(byte)의 치환(substitution)이 이루어지며, 복호화시는 S-box의 $GF(2^8)$상의 역원에 의하여 치환된다 본 연구에서는 속도의 향상을 위하여 $GF(2)$상에서 affine transform을 직접 구현하지 않고, S-box와 역 S-box의 처리 결과를 테이블(table)로 저장한 뒤 참조(lookup)하도록 하였다.

②**ShiftRow/InvShiftRow**는 암호화 경우에 행(Row)의 인덱스(index)의 값에 따라서 0,1,2,3 만큼 Left Shift Rotate하도록 한다. 복호화 경우 역과정을 거쳐 복호화 한다.

③**MixColumn/InvMixColumn**은 한 워드(word,32bit) 단위의 열(Column)에 대해서 [그림 6]에 보인 행렬 c와 c^{-1}의 곱셈을 수행한다. 즉, 암호화 및 복호화의 경우에 $GF(2^8)$상의 한 바이트는 다음 두식에 의하여 연산이 이루어진다:

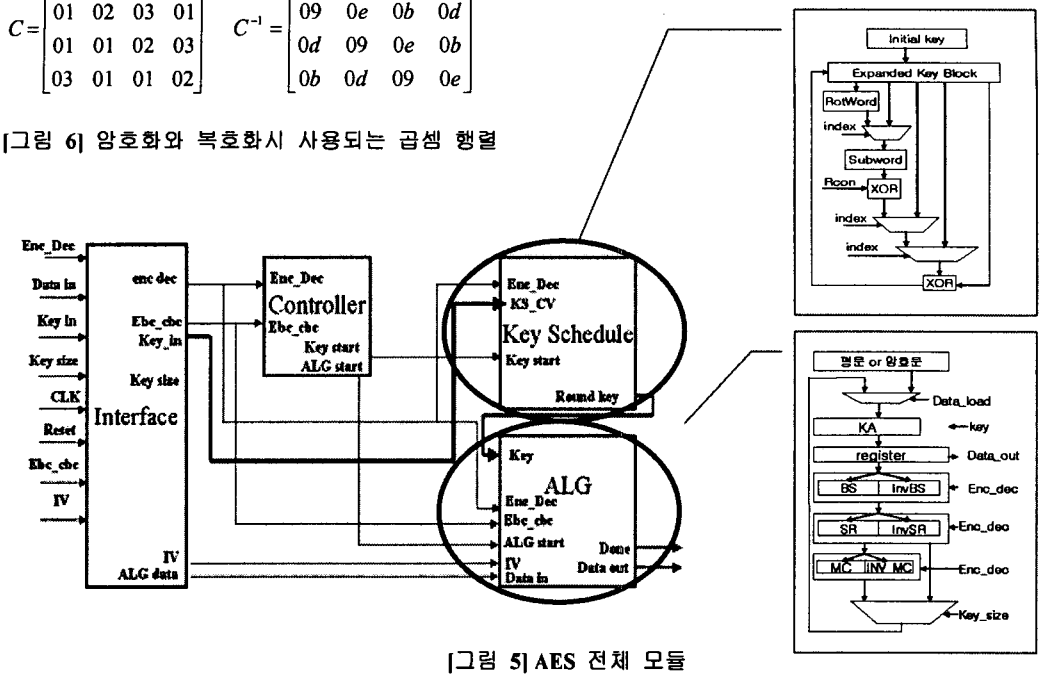
암호화 : $d_i(x)=c_{i1}\cdot x_1+c_{i2}\cdot x_2+c_{i3}\cdot x_3+c_{i4}\cdot x_4$

복호화 : $e_i(x)=c_{i1}^{-1}\cdot x_1+c_{i2}^{-1}\cdot x_2+c_{i3}^{-1}\cdot x_3+c^{-1}{}_{i4}\cdot x_4$

여기서 $c\cdot x$는 $GF(2^8)$상에서의 곱셈을 의미한다.

$$C=\begin{bmatrix}02 & 03 & 01 & 01\\ 01 & 02 & 03 & 01\\ 01 & 01 & 02 & 03\\ 03 & 01 & 01 & 02\end{bmatrix} \quad C^{-1}=\begin{bmatrix}0e & 0b & 0d & 09\\ 09 & 0e & 0b & 0d\\ 0d & 09 & 0e & 0b\\ 0b & 0d & 09 & 0e\end{bmatrix}$$

[그림 6] 암호화와 복호화시 사용되는 곱셈 행렬



[그림 5] AES 전체 모듈

$GF(2^8)$ 상에서 곱셈을 수행하는 과정의 구현은 다소 복잡한 과정을 거치게 된다. 본 논문에서는 이 곱셈연산을 위하여 두가지 방법으로 구현 후 성능을 비교하였다.

1) <u>직접 곱셈 로직을 구현하는 방법</u> : 첫번째는 암호화나 복호화 과정에서 발생하는 모든 곱셈 연산을 직접 하드웨어 로직으로 구현하는 방법이다. 예를 들어, $02 \cdot x$는 x를 Left Shift하고 {1B}와 XOR를 선택적으로 수행하도록 구현한다[11]. 여기서는 {02} 이외의 다른 수인 {03}, {09}, {0B}, {0D}, {0E} 등과의 곱셈을 다른 로직으로 구현하였다.

2) <u>대수적 성질을 이용한 테이블을 참조 방법</u> : 첫번째 방법으로 구현할 경우의 문제점은 암호와 및 복호화를 위하여 서로 다른 모듈로 설계되어야 하는 점이다. 본 논문에서는 암호화와 복호화를 동일한 모듈로 구성하고 있으므로, 두번째 방식에서는 (식-1)과 같은 대수적 성질을 이용한다.

$$ab = c^{\log_c ab} = c^{\log_c a + \log_c b} \qquad (식\text{-}1)$$

여기서, c 는 $GF(2^8)$ 상에서의 임의의 생성자(generator)이다.

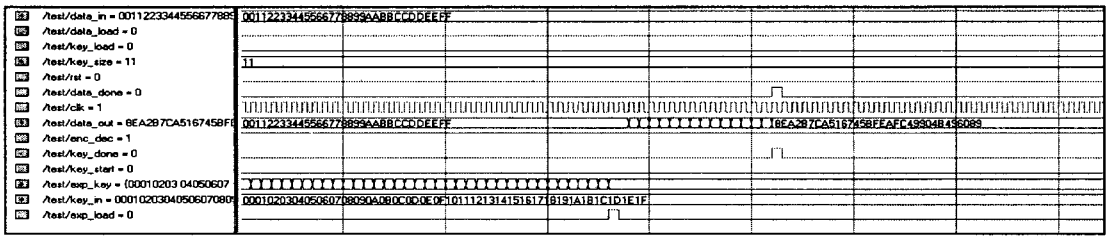
즉, $f(x) = \log_c(x)$와 $g(x) = c^x$를 미리 계산하여 테이블로 구성한다. 곱셈의 계산은 log 테이블을 참조하여 덧셈을 수행하고, 지수 테이블을 참조하여 결과를 얻는 방법[12]으로 구현하였다.

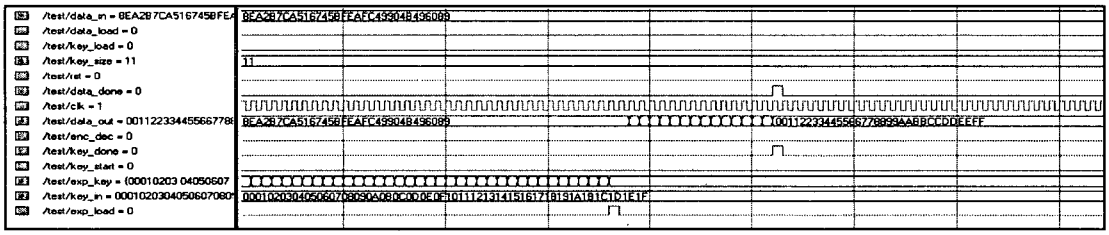
④**AddRoundkey**는 ①,②,③의 수행결과로 나온 계산 결과와 확장된 키를 단순히 Bitwise-XOR를 수행한다.

4. 성능 분석 및 결론

본 논문에서는 AES를 하드웨어 설계 언어인 VHDL로 구현하였다. 구현된 VHDL 코드는 Model Technology 사의 ModelSim5.2c를 가지고 시뮬레이션 되었으며, Altera 사의 Synplify Pro 7.0.1에서 로직 합성(logic synthesis) 하였다.

[암호화]





[그림 7] 키의 크기가 256비트의 암호와 복호화 시뮬레이션

[그림 7]은 키의 크기가 256 비트일 때, 암호와 과정과 복호화 과정의 시뮬레이션 결과를 보인 것이다. 키의 크기가 128 및 196 비트의 경우에도 올바른 결과를 얻었다.

설계된 AES 알고리즘에서 암호화와 복호화에 소모되는 클럭(clock)의 수는 각 키의 크기 128, 196, 256 비트에 따른 라운드 수와 관련하여 각각 10클럭, 12클럭,14클럭이 소모되었고, 50Mhz 에서 각각 640Mbits/s, 533.3Mbits/s, 457.1Mbits/s의 성능을 보였다.

본 논문에서는 AES의 구현시 단점을 보완하여 암호화/복호화 단계가 동일한 구조로 동작하도록 구현되었다. 이를 위하여 **MixColumn/InvMixColumn** 모듈을 두가지 방식으로 구현하여 실험하였고, 비슷한 성능을 보임을 확인하였다. 또한, AES 표준에 따라 키의 크기가 128, 196, 및256 비트의 경우에 동일하게 구현되어 효율성을 높이도록 설계하였다. 그러므로, 본 논문에서 구현된 모듈은 단순한 구조를 가지며, FPGA 탑재시 효율적으로 공간을 사용한다.

앞으로의 연구는 현재 부분적으로 구현 중인 IPv6용 IPSec의 전체 모듈을 효율적으로 구현하고 성능을 분석하여, 상용화가 가능한 IPSec 하드웨어 칩을 개발하는데 목적이 있다.

5. 참고 문헌

[1] http://www.ietf.org/html.charters/ipv6-charter.html
[2] IP Security Protocols (ipsec) Charter – Latest PFCs and Internet Drafts for Ipsec, http://www.ietf.org/html.charters/IPSec-charter.html
[3] Advanced Encryption Standard Development Effort, http://www.nist.gov/aes
[4] RFC2402: IP Authentication Header (AH)
[5] RFC2406: IP Encapsulating Security Payload (ESP)
[6] L.L. Peterson and B.S. Davie, "Computer Networks - A Systems Approach", Morgan Kaufmann, 2001
[7] Wiznet, USB 카메라용 인터넷어댑터 설계 결과 보고서, Manuscript, 2001.
[8] B. Weeks, M. Bean, T. Rozylowicz, and C. Ficke, "Hardware Performance Simulations of Round 2 Advanced Encryption Standard Algorithms", National Security Agency
[9] P. Chodowice, K. Gaj, P. Bellows, and B. Schott, "Experimental Testing of the Gigabit IPSec-Compliant Implementations of Rijndael and Triple DES Using SLAAC-1V FPGA Accelerator Board", Proc. Information Security Conference, October, 2001
[10] Draft NIST Special Publication 800-17, "Modes of Operation Validation System(MOVS) Requirements and Procedures", May, 1996.
[11] 이종수, 염동복, 박종서, "차세대 표준 암호 알고리즘(AES) RIJNDAEL의 하드웨어 구현," ITRC Forum pp. D1-21, 2001
[12] P. Barreto, V. Rijmen: "Rijndael ANSI C Reference Code", NIST, October, 2000