무결성 검증 기능을 제공하는 가상사설망 구현에 관한 연구

박성준, 김창배, 김창수

부경대학교 컴퓨터멀티미디어공학부

초 록

인터넷의 급속한 보급과 공용망을 통한 전자상거래 등의 활용이 늘어나면서 무결성 보장과 보안장치의 필요성이 부각되고 있다. 무결성 보장을 위하여 여러 가지 방법들이 사용되고 있으며 특히 해쉬함수를 이용한 메시지 인증 방법이 다양한 형태로 응용되어 사용되고 있다. 최근 공용망의 보안 취약성을 개선하여 사설망과 같은 환경을 구성하기 위해 가상사설망(VPN)의 개념이 제안되었다. 특히 IP계층에서 VPN을 구성하는 표준으로 IETF에서 제안된 IPsec은 암호화와 무결성기능을 포함한 VPN구성 방법으로 최근 활발히 연구되고 있다. 본 논문에서는 IPsec에 따라 실제로 호스트간의 통신 암호화와 해쉬함수를 이용한 무결성 검증 기능을 제공하는 VPN을 구현하였다.

Ⅰ. 서론

오늘날 기업에서는 외부 네트워크와 정보교환에 관한 요구와 재택 근무자의 증가로 인하여 네트워크의 범위가 확대되어 감에 따라 상호간에 안전한 통신을 하기 위해 사용해오던 전용망에 대한 투자 비용과 그에 따른 운영과 관리가 커다란 문제로 되고 있다. 따라서, 이에 대한 해결책으로 제안된 가상사설망(VPN)이 주목을 받고 있다.

가상사설망은 공중망 환경에서 보안의 취약성을 개선하여 사설망과 같은 환경을 구성하는 것으로, 암호화와 인증 서비스를 이용하여 소프트웨어적으로 구현되어 기존의 전용망에 비해 비용이 저렴하고 설치 및 유지보수가 쉽다. 앞으로 인터넷을 통한 전자상거래는 급속도로 성장하게 될 것이고 그와 함께 가장 중요한 문제로 부각되는 것은 통신의 보안 문제와 전송 데이터의 무결성 보장 문제이다. 따라서 VPN의 수요와 가치는 앞으로 매우 높아질 것으로 기대된다.

본 논문에서는 위와 같은 필요성과 상황에 기인하여 실제로 가상사설망을 구현해보고자 하였다. 가상사설망의 구현을 프로토콜 계층별로 나누어 볼 때 응용계층, 트랜스포트 계층, 링크계층으로 나누어 볼 수 있는데, 본 논문에서는 IETF에서 IP계층 보안의 표준안으로 제시한 IPsec에 따라 IP 계층에서 암호화와 인증 기능을 추가하였다. 특히 IPsec의 ESP 구현 방법 중 Transport mode를 이용하였으며 AH 구현을 위해 MD5 해쉬 알고리즘을 이용하여 호스트간의 VPN을 구현하고자 하였다.

본 논문의 구성은 2장에서 관련연구로 IP계층에서 보안 프로토콜로 표준화되어 가는 IPsec과 MD5 해쉬 알고리즘에 대해 살펴본다. 3장에서는 2장에서 살펴보았던 IPsec이용하여 실제로 호스트간의 VPN을 구현하고 구현된 VPN에서의 해쉬함수의 사용 방법을 제시한다. 마지막으로 4장에서 결론 및 향후 연구과제에 대해서 살펴본다.

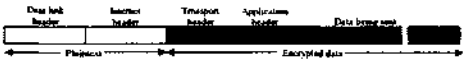
Ⅱ. 관련 연구

2.1 IPsec(IP Security)

IPsec은 TCP/IP 통신을 보호하기 위한 다목적 프로토콜의 한 형태이며, 링크계층의 암호화를 포함하여 다른 기술들과 구별되는 특징을 가지고 있다. 실제로 이것은 주어진 호스트 상에서 사용자 사이가 아니라 호스트들간의 트래픽을 보호하는데 적합하다. 또한 IPsec은 어떤 서비스나 응용프로그램도 보호할 수 있으며 원격 사용자, 라우터, 침입차단시스템 등을 수정하지 않고도 구현할 수 있다는 장점이 있다. IPsec은 네트워크 계층에서의 가

상사설망 구현을 위해서 무결성과 인증을 보장하는 인증헤더(Authentication Header)와 암호화된 캡슐화를 통하여 비밀성을 보장할 수 있는 ESP(Encapsulation Security Payload)를 사용한다. 인증 헤더는 패킷 안의 데이터를 검증 가능한 서명과 결합시켜 데이터 송신자의 신원을 확인하고 데이터가 변하지 않았음을 검증할 수 있도록 하며, ESP는 각 패킷의 데이터를 네트워크 상에서 불법적으로 해독할 수 없도록 한다. 또한 키 관리 메커니즘으로 다양한 방법이 사용될 수 있으며, IEFT의 제안에서는 ISAKMP를 사용한다.

[그림 2.1]에서 보는 바와 같이, IPsec은 Snooping 혹은 변조로부터 IP 패킷을 보호하기 위해 현재의 IP 프로토콜을 확장시킨 것이다. IPsec은 새로운 IPv6 개발의 한 부분으로서 발전되어 왔고 현재의 IPv4에서도 함께 동작할 수 있도록 적용되어 왔다.



[그림 2.1] IPsec 프로토콜의 기본 패킷 포맷

2.1.1 인증헤더(Authentication Header : AH)

인증헤더는 IP 데이터그램을 인증하기 위해 필요한 정보를 포함하는 방법으로 보안효과 특히 인증과 무결성을 보장해 주는 메카니즘이다. 기본적인 개념은 IP 데이터그램 전체에 걸쳐서 단방향 해쉬함수를 이용하여 인증 정보를 추가함으로써 보안을 제공한다. 인증헤더의 필드 구성은 [그림 2.2]와 같고, 위치는 IPv6에서는 Fregmentation과 End-to-End 다음에, TCP 헤더 앞에 위치하고, IPv4에서는 IP헤더 바로 다음에 위치한다([그림 2.3]참조). 인증헤더를 적용한 패킷의 처리순서는 우선 목적지와 사용자 ID를 기초로 하여 만든 SA(Security Association)를 선택하고 인증데이타를 계산하고 인증헤더를 완성한다. 수신지에서는 목적지 주소를 가지고 SA를 선택하고 수신패킷의 변화여부를 체크한다.

Next Header(8)	Payload Length(8)	Reserved(16)
SPI(Security Parameters Index)(32)		
Sequence Number(32)		
Authentication Data(32*n)		

[그림 2.2] AH 헤더의 필드 구성

IPv6 헤더	Hop-by-Hop or Routing	AH 헤더	Others	Upper Layer Protocol

IPv4 헤더	AH 헤더	Upper Layer Protocol

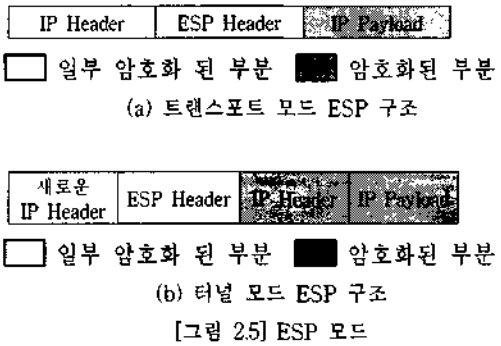
[그림 2.3] IPv4와 IPv6에서의 인증헤더 위치

2.1.2 ESP(Encapsulation Security Payload)

ESP는 암호화 기법을 사용하여 데이터의 재전송 방지, 비밀성의 기능을 제공하는 프로토콜이다. 기본적인 개념은 보호될 데이터를 암호화하여 ESP헤더의 데이터 영역에 삽입하고 IP헤더에 ESP 헤더를 추가하는 것이다. ESP헤더의 구성 필드는 [그림 2.4]와 같고, 이때 보호되는 영역이 트랜스포트 세그먼트인지 IP 데이타그램 전체인지에 따라 트랜스포트 모드와 터널링 모드로 나누어진다 ([그림 2.5] 참조).

SPI(Security Parameters Index)(32)		
Sequence Number (32)		
Payload Data		
Padding(0-255)	Pad Length(8)	Next Header(8)
Authentication Data(32*n)		

[그림 2.4] ESP 헤더의 구성 필드



[그림 2.5] ESP 모드

2.1.3 SA(Security Association)

SA는 인증헤더와 ESP 헤더를 처리하기 위해 필요한 정보를 자료구조로부터 찾아내기 위한 식별자를 의미한다. 송신호스트는 자신의 사용자 식별자와 목적지 주소를 사용하여 적당한 SA를 선택하고 수신호스트에서는 인증헤더와 ESP 헤더에 있는 SPI, 목적지 주소 등에 따라 해당하는 SA를 선택하게 된다. 이러한 SA는 단방향 접속을 위한 것이므로 전형적인 두 종단간의 양방향 통신을 위해서 두 개의 SA를 필요로 한다. SA와 관련된 매개변수들을 포함하는 데이터베이스로는 보안정책 데이터베이스(SPD)와 SA 데이터베이스(SAD)의

두 가지가 있는데 SPD는 IP 데이타그램에 어떤 서비스가 어떤 형태로 제공되는지 명시하고, SAD의 엔트리에는 해당하는 SA에 관련된 매개 변수들이 정의되어 있다.

2.2 해쉬함수

해쉬함수는 중요 정보의 무결성 확인과 메시지 인증 코드(Message authentication code : MAC), 디지털 서명(digital signature)의 효율성 증대 등의 목적으로 임의의 길이의 비트 스트링을 입력으로 받아 고정된 짧은 길이(주로 128bit 혹은, 160bit)의 비트 스트링을 출력하는 함수이다. 이 출력은 흔히 해쉬값, 메시지 다이제스트(message digest) 또는 fingerprint 등으로 불린다.

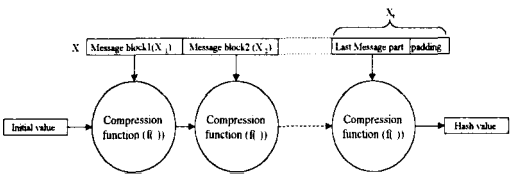
2.2.1 일반적인 해쉬함수의 구조

일반적인 해쉬함수는 고정된 길이의 입력을 가진 압축함수에 기반하는 반복적인 구조를 가지며 수식적으로 다음과 같이 표현한다.

$$H_0 = IV,\ H_i = f(X_i, H_{i-1}),\ i = 1,2,\ldots,t\ ,\ h(X) = H_t$$

위 식에서 $X_1 \sim X_t$는 입력문을 수행단위로 나눈 각 블록을 의미하고 입력정보는 t개의 b-bit블록 X_1에서 X_t로 나뉘고 전체 비트 수가 블록 길이 b의 배수가 아니면, 채워 넣기(padding)절차가 수행되어야 한다. 해쉬함수 $h()$는 라운드함수 또는 압축함수인 $f()$로 구성된다. H_i는 n-bit 길이의 연쇄변수이고 IV는 초기값(Initial Value), $h(X) = H_t$ 는 해쉬함수의 결과이다.

해쉬값의 계산은 연쇄변수에 의존한다. 해쉬 계산을 시작할 때, 이 연쇄변수는 알고리즘의 일부로 명시된 고정된 초기 값을 가진다. 압축함수는 해쉬되어질 메시지 블록을 입력으로 받아 이 연쇄변수의 값을 갱신한다. 이 과정이 모든 메시지 블록에 대해 순환적으로 반복되고, 연쇄변수의 마지막 값이 그 메시지에 대한 해쉬값으로 출력된다([그림 2.6]참고).

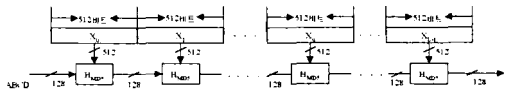


[그림 2.6] 반복적인 해쉬함수의 일반적인 구조

2.2.2 MD5

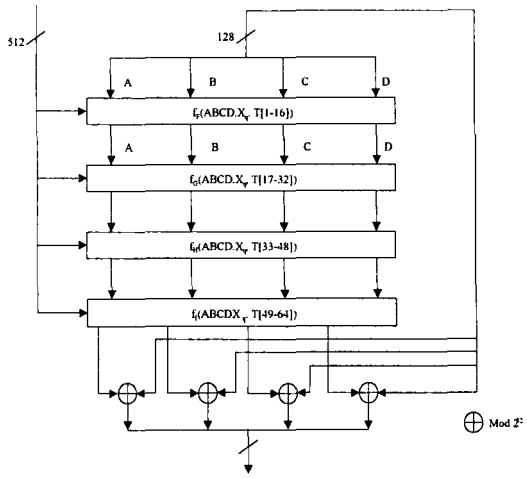
Merkle과 Damgard의 이론에 기반을 둔 반복적인 형태의 전용 해쉬함수로 1990년 Rivest가 "The MD4 message-digest algorithm"[RFC 1320]에서 제안하였다. MD5 알고리즘은 1990년 MD4가 발표된 이후 MD4에 대한 많은 공격 시도들이 성공하면서 MD4의 안정성이 문제가 되자 1991년 Rivest에 의해 MD4의 강화된 형태로 발표되었으며 현재 많은 MD계열의 해쉬함수들의 설계 기초가 되고 있으며 국내·외에서 가장 많이 사용되는 해쉬함수이다.

MD5 알고리즘은 입력 메시지를 512의 배수가 되도록 두 단계의 패딩(64 비트의 원래 메시지 길이 포함)과정을 거친 후 기본적으로 512비트의 메시지 블록(16개의 메시지 워드)과 32비트 워드를 입력으로 하여 4개의 워드를 출력하는 압축 함수를 반복적으로 적용하여 128비트의 해쉬 결과(message digest)를 출력한다([그림2.7] 참고).



[그림 2.7] MD5의 해쉬과정

압축 함수는 [그림 2.8]과 같이 4라운드로 구성되어 있으며, 각 라운드는 16단계로 이루어진다.



[그림 2.8] H_{MD5} 처리과정

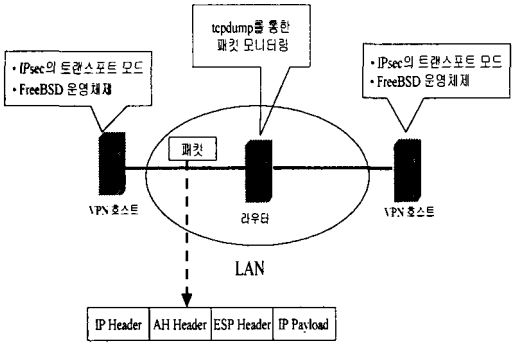
Ⅲ. IPsec을 이용한 가상사설망 구현

IPsec을 이용하여 가상사설망을 구현하는 것은 종단간에 암호화와 인증기능을 부여하는 것이라고 할 수 있다. 종단이되는 호스트가 게이트웨이나 라우터인 경우 터널링 모드를 사용하여 두 호스트간

외 터널을 구성하고, 그렇지 않은 경우 트랜스포트 모드를 사용한다. 본 논문에서는 호스트간에 트랜스포트 모드를 사용하여 암호화와 인증 기능을 부여하고자 하였다.

3.1 구현환경

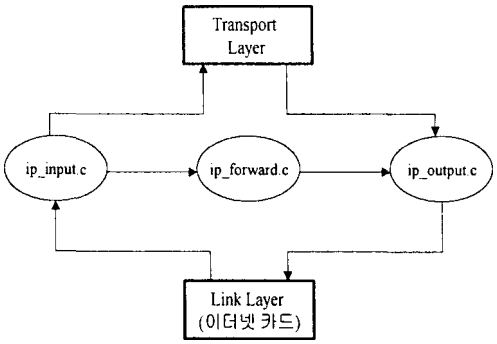
IPsec은 기존의 TCP/IP 통신 프로토콜의 IP계층에 보안기능을 추가한 프로토콜이다. 이 IPsec 프로토콜의 ESP와 AH의 구현을 위해서는 운영체제가 TCP/IP 통신 프로토콜을 지원하고, 커널 수정이 가능해야 한다. 그래서 본 논문에서는 커널 소스가 공개된 Free-BSD(v 2.2.7)를 시스템의 운영체제로 사용하였다. 그리고 VPN이 정확히 구현되었는지 확인하기 위해 VPN이 적용된 호스트간의 패킷을 모니터링 할 수 있는 tcpdump를 라우터에 설치하였다([그림 3.1]참고).



[그림 3.1] VPN의 구현 환경

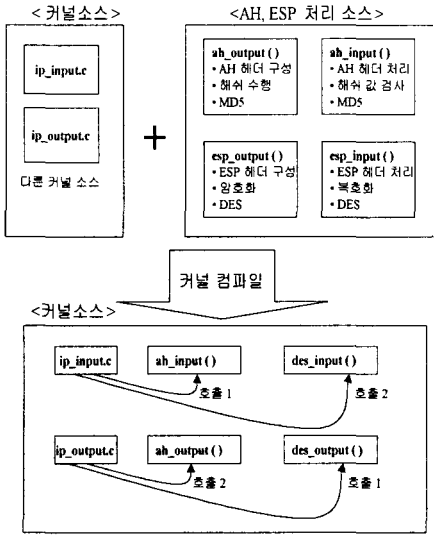
3.2 구현내용

구현내용을 살펴보기에 전에 우선, BSD 커널의 IP패킷 처리과정을 살펴보면 [그림 3.2]와 같다. 이더넷 카드를 통해 들어온 패킷은 ip_input.c에서 우선 목적지 주소를 검사한다. 만약 목적지 주소가 자신의 주소와 일치하면 상위 계층으로 패킷을 넘겨주고, 그렇지 않으면 ip_forward.c를 따라 ip_output.c로 보내져 일반적인 출력 패킷과 마찬가지로 네트워크로 나가게 된다. 추가적으로 ip_output.c는 출력 패킷에 대해 옵션 처리, 경로설정, Fragmentation등의 처리를 하고, ip_input.c는 입력 패킷에 대해 IP헤더 검증, 체크섬 검사, 포워딩, 옵션처리, Defragmentation등의 기능을 수행한다.



[그림 3.2] IP계층의 패킷 처리 과정

IP 계층에서 암호화와 인증 기능의 구현은 크게 출력 패킷에 대한 처리와 입력 패킷에 대한 처리 과정으로 나누어 볼 수 있다. 처리과정을 살펴보면 Free-BSD의 커널 내용 중 IP 패킷을 처리하는 ip_input.c와 ip_output.c 파일에 ESP와 AH 처리 루틴의 호출함수를 추가하고, 처리 루틴을 커널에 포함시켜 커널을 재 컴파일 하게된다([그림 3.3] 참조). 여기서 기밀성 보장을 위해 ESP에 사용된 암호 알고리즘은 비밀키 암호방식의 DES이고, 인증을 위해 AH에서는 단방향 해쉬함수인 MD5를 이용한 HMAC_MD5를 사용하였다.



[그림 3.3] VPN의 구현

3.2.1 출력 패킷에 대한 처리(ip_output.c)

IPsec을 적용하기 위해서 우선 출력 패킷의 목적지 주소가 VPN 호스트인가를 판단한다. 만일 VPN 호스트이면 ESP와 AH처리 루틴을 수행하고 그렇지 않으면 원래 패킷 처리대로 진행한다. IPsec에 따르면 목적지 주소가 VPN 호스트인지 그리고 ESP와 AH처리를 위한 키 등의 설정을 위해 데이터 베이스를 구축하여야 하지만 목적이

VPN의 구현이므로 키의 설정은 임의적으로 설정해놓고 VPN 호스트인지는 목적지 주소를 VPN이 적용된 호스트의 IP주소와 비교한다. 출력 패킷의 AH와 ESP의 주요 처리 과정은 [그림 3.4]와 같다.

```
if(!bcmp(addr_dst, "203.247.166.109", 15))          … ①
{ if(h_len > sizeof(struct ip))                     … ②
  { ip_stripoptions(m, (struct mbuf *)0);
    h_len = sizeof(struct ip);
  }
 ip->ip_len = htons((u_short)ip->ip_len);           … ③
 ip->ip_off = htons((u_short)ip->ip_off);
 ip->ip_sum = 0;
 ip->ip_sum = in_cksum(m,h_len);
 esp_output(m, &m_imsi, des_p);                     … ④
 m=m_imsi;                                          … ⑤
 ah_output(m, &m_imsi, ah_key);                     … ⑥
 m = m_imsi;                                        … ⑦
 ip->ip_sum = 0;
 ip->ip_sum = in_cksum(m, h_len);                   … ⑧
 ip = mtod(m, struct *ip);
 NTOHS(ip->ip_len);                                 … ⑨
 NTOHS(ip->off);
}
```

[그림 3.4] 출력 패킷의 처리과정

① 문자열 형태의 목적지 IP주소와 VPN을 설정한 호스트의 IP주소가 일치하는지 판별
② 만약 옵션이 있으면 제거
③ IP 해더의 ip_len, ip_off, ip_sum을 새롭게 계산
④ ESP 처리함수를 호출하는 부분. 함수 호출시 인자는 메모리버퍼, 작업버퍼, ESP 처리를 위해 사용되는 변수의 포인터이다.
⑤ ESP 수행결과를 메모리버퍼에 저장
⑥ AH 처리함수를 호출하는 부분. AH 헤더를 만들어서 IP 데이터그램의 IP 헤더 다음에 끼워넣은 다음 IP 데이터그램의 전체에 걸쳐서 해쉬함수를 적용하여 메시지 인증값을 구하여 AH 헤더의 Authentication Data 필드에 삽입하게 됨.
⑦ AH 수행 결과를 메모리버퍼에 저장
⑧ IP 체크섬의 재 계산
⑨ AH와 ESP처리를 위해 네트워크 바이트 순서로 변경되었던 IP패킷을 호스트 바이트 순서로 변경

3.2.2 입력 패킷에 대한 처리(ip_input.c)

출력 패킷에 대한 처리과정과 마찬가지로 우선 송신지 주소가 VPN 호스트인지 판단하고 추가적으로 이루어져야 하는 작업은 IP헤더의 프로토콜 번호를 보고 AH인지 ESP인지를 판단한다. 만약 수신된 패킷이 AH와 ESP를 적용한 상태라면 먼저 AH처리 루틴을 수행하고 해쉬함수의 수행결과 패킷의 변조가 없음이 인증되면, ESP처리 루틴을 수행하여 복호화 하게 된다. 입력 패킷의 주요 처리 과정은 [그림 3.5]와 같다.

```
if(!bcmp("addr_src", "203.247.166.112", 15))        … ①
{ if(ip->ip_p == 51 || ip->ip_p == 50)
   { if (ip->ip_p == 51)
     {  if(h_len > sizeof(struct ip)                … ②
        { ip_stripoptins(m, (struct mbuf *)0);
          h_len = sizeof(struct ip);
        }
        if(ah_input(m, &m_imsi, ah_key))
        goto error;
        m=m_imsi;                                   … ③
     }
    if(ip->ip_p == 50)                              … ④
     { if(h_len > sizeof(struct ip)
        { ip_stripoptins(m, (struct mbuf *)0);
          h_len = sizeof(struct ip);
        }
       m = esp_input(m, des_p);                     … ⑤
       m = m_imsi;                                  … ⑥
     }
   }
}
```

[그림 3.5] 입력 패킷에 대한 처리과정

① 문자열 형태의 송신지 IP 주소와 VPN을 설정한 호스트의 IP 주소가 일치하는지 판별
② AH를 적용한 경우, 들어온 패킷에 대해 옵션이 있으면 제거하고, 해쉬함수를 수행하여 변조 여부를 파악하여 변조된 경우는 패킷을 버리고, 변조되지 않은 경우에는 AH헤더를 제거한다.
③ AH 처리 결과를 메모리버퍼에 저장
④ ESP를 적용한 경우, 옵션이 있으면 옵션부터 제거
⑤ ESP 구현 함수 호출하는 부분. 복호화 과정을 수행하고 ESP 헤더 제거
⑥ ESP 처리 결과를 메모리버퍼에 저장

3.2.3 구현 결과 분석

본 논문에서는 Free-BSD의 커널 수정을 통하여 IP 패킷에 암호화와 인증 기능을 구현하였다. 올바르게 구현되었는지 검사하기 위해 VPN 호스트간에 전송되는 패킷을 tcpdump를 사용하여 모니터링

해보았다. 우선 AH헤더와 ESP헤더가 IP헤더에 추가되었는지는 전송 패킷의 프로토콜번호를 확인하여 각각 51과 50번임을 확인하는 방법으로 검증하여 올바르게 구성되었음이 밝혀졌으며, AH가 올바른 해쉬 수행과정을 거치는지를 검증하기 위해 실제로 나가는 패킷에 대해 AH와 ESP헤더의 처리과정을 거친 후 전송직전에 IP패킷의 임의의 위치를 변경시켜 전송하여 수신측에서 패킷 변조사실을 검출할 수 있는지를 테스트하여 무결성 검사 기능의 정상적인 작동여부를 증명하였다([그림 3.6 참조]).

```
addr_dst = inet_ntoa(ip->ip_dst);
addr_src = inet_ntoa(ip->ip_src);                    … ①
if(!bcmp(addr_dst,"203.247.166.109",15)              … ②
 &&!bcmp(src,"203.247.166.112",15))
  { modified = m->data + sizeof(struct ip);          … ③
    *modified = *modified + 변조위치;
  }
```

[그림 3.6] 패킷 변조 과정

① 송신지와 목적지의 주소를 문자열 형태로 변환
② 송신지와 목적지의 주소가 VPN을 적용한 호스트인지 판별
③ 메모리버퍼의 IP 헤더를 제외한 IP 데이터그램의 임의의 위치를 변경

Ⅳ. 결론 및 향후 과제

VPN은 원격사이트들이 서로 안전하게 접속하기 위해 상대적으로 비용이 저렴하고 광범위한 공용망을 이용하는 기술이다. 특히 IPsec프로토콜은 기존의 응용프로그램이나 다른 프로토콜의 변경 없이 트래픽을 보호할 수 있는 장점을 가진다. 본 논문에서는 IPsec을 이용하여 IP계층에서 상위 계층에 투명성을 유지하면서 데이터 비밀성과 메시지 인증 기능을 제공하고자 하였다. 물론 완전한 VPN을 구성하려면 키 관리나 SA구현 등 보다 복잡한 문제들이 남아있으며, ESP와 AH에서 사용하는 암호화와 해쉬함수들의 종류를 추가하여 사용 알고리즘의 선택의 폭을 넓혀야 하는 문제들이 남아있으나, 본 논문의 구현내용만 가지고도 엄격한 보안이 요구되지 않는 호스트간의 통신에 보안기능을 제공해 줄 수 있을 것이다. 그리고 AH와 ESP처리과정에서 부가적인 오버헤드가 FTP를 통해 전송시 평문의 최고 30%까지 생길 수 있다고 하는데, 하드웨어적 구현 등을 통해 어느 정도는 오버헤드를 줄일 수 있을 것이다. 본 논문에서는 정적 IP를 가진 호스트간에 보안기능을 부여하고자 하였다. 앞으로의 연구과제는 동적으로 IP를 갖는 이동형 시스템까지 확장해 보고자 한다.

[참고문헌]

[1] Gary R. Wright, W. Richard Stevens "TCP/IP Illustrated, Volume 1, 2" ,1994

[2] W. Richard Stevens "UNIX Network Programmming, Volume 1", 1998

[3] [Arch] Kent, S., and R. Atkinson, "Security Architecture for the Internet Protocol", RFC 2401, November 1998.

[4] [AH] Kent, S., and R. Atkinson, "IP Authentication Header", RFC 2402, November 1998.

[5] [ESP] Kent, S., and R. Atkinson, "IP Encapsulating Security Payload (ESP)", RFC 2406, November 1998.

[6] [Thayer97] R. Thayer, "IP Security Document Roadmap", RFC 2411, November 1998.

[7] [ISAKMP] D. Maughan, "Internet Security Association and Key Management Protocol", RFC 2408, November 1998

[8] [Oakley] Orman, H., "The Oakley Key Determination Protocol", RFC 2412, November 1998

[9] 한국정보보호 센터, "IP계층과 응용계층에서의 VPN 보안기술 표준(안)", November, 1998.

[10] [MD5] Rivest R., "The MD5 Message-Digest Algorithm", RFC 1321, April 1992.

[11] [HMAC] H.Krawczyk, "Keyed-Hashing for Message Authentication" RFC 2104, February 1997

[12] [DES] P. Karn, P. Metzger, W. Simpson "The ESP DES-CBC Transform" RFC 1829, August 1995

[13] 이경현, 신상욱, 신원, 김혜정, 이인실, 심경섭 "해쉬 알고리즘 공격 방법 및 안전성 분석 기법" 한국전자통신 연구원 최종 보고서, 1998, 11

[14] 원동호 "현대 암호학" http://203.252.53.61/dhwon/dhwon_k2.html

[15] 신상욱, 류대현, 이상진, 이경현, "MD_x 계열 해쉬함수에 기반한 새로운 해쉬함수", 정보처리학회 추계학술발표 논문집, vol.4, No.2, pp. 1354-1359, 1997.