2002년 한국정보처리학회 춘계학술발표논문집 제9권 제1호

블록 암호 알고리즘을 이용하는 IP 패킷 암/복호화 모듈의 설계

정준목
LG 엔시스
e-mail : chunmok@lgnsys.com

Design of IP Packet Encryption/Decryption Module using Block Cipher Algorithm

Chun-Mok Chung
R&D Division, LG N-Sys

요 약

중간 노드들에서 통신 데이터를 불법으로 획득하는 스니핑에 대한 대안으로 암호화 통신에 대한 요구가 증가하고 있다. 이를 위해 본 논문에서는 암호화 통신을 위해 블록 암호 알고리즘을 사용한 패킷 암/복호화 모듈을 제시하고, SEED 알고리즘을 이용하여 리눅스에 구현한 사례를 기술한다. 이 모듈은 기존의 네트워크와 응용 프로그램에 영향을 주지 않고 암호화 통신 기능을 제공하기위해 IP 패킷의 헤더 정보를 변경하는 방법을 사용한다.

1. 서론

인터넷이 발전하고 더 많은 시스템들이 네트워크로 연결되고 있다. 기업의 중요한 시스템들도 더욱 높은 활용성과 접근성을 위해 네트워크로 연결되어지고 있다. 이들은 지역적으로 떨어진 지점이나 업무상 이동하는 직원들에게 개방될 목적으로 네트워크에 연결되어 있지만, 인터넷을 중간 매체로 사용하는 구조적 특성상 의도하지 않은 외부의 사용자들에 의해 기업의 정보들이 외부로 유출될 위험을 가지게 되었다. 기업의 두 지점간의 통신 데이터가 인터넷의 다양한 노드들을 경유하므로, 이들 중간 노드들에서 스니핑 기법을 사용하여 지나가는 통신 데이터를 불법으로 획득할 수 있다. 이러한 스니핑에 대한 대안으로, 통신 데이터를 암호 알고리즘으로 암호화하여 통신하는 암호화 통신에 대한 요구가 증가하고 있고, 이를 위해 다양한 암호 알고리즘과 암호화 통신 모델들이 제안되고 있다.

본 논문에서는 암호화 통신을 위해 블록 암호 알고리즘을 사용한 패킷 암/복호화 모듈을 제시한다. 이 모듈은 기존의 네트워크와 응용 프로그램에 영향을 주지 않고 암호화 통신 기능을 제공하는 것을 목적으로 한다.

2. 요구 사항과 접근 방법

암호화 통신 기능은 기존에 이미 구축되어 운영되는 시스템에서 통신 데이터를 보호하기위해 사용된다. 이들 기존 시스템을 위한 암호화 통신 기능을 설계하는 우리에게 주어진 세가지 주요 요구 사항은 다음과 같다.

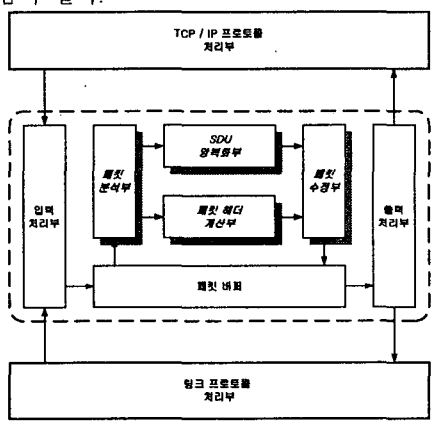
먼저, 기존의 통신 프로그램(telnet, ftp, email 등)을 수정 없이 사용할 수 있어야 한다. 암호화 통신을 위해 기존의 통신 프로그램을 수정해야 한다면, 이는 비용이나 시간의 증가 뿐만 아니라 새로운 통신 프로그램의 사용에 따른 안정성의 문제도 발생할 수 있다. 두번째로, 원거리에 떨어져있는 두 노드가 통신을 하기 위해서는 중간에 존재하는 라우터나 허브와 같은 다양한 통신 장비들이 두 노드간의 패킷을 상호 전달해서 이루어진다. 이들 장비는 기존의 TCP/IP 표준 프로토콜을 사용하므로, 암호화 통신이 표준 TCP/IP 프로토콜을 사용하는 중간 노드의 장비들에게 영향을 주거나, 그들로부터 영향을 받아서는 안된다. 이들 중간 노드의 장비들은 자신의 소유가 아니거나 수정 권한이 없기 때문에 암호화 통신을 위해 수정이 불가능할 수 있다. 끝으로, 암호 알고리즘으로 국가 표준 암

호 알고리즘(SEED)[1]와 같은 공인된 블록 암호 알고리즘을 지원해야 한다. 관공서나 금융 기관과 같은 중요 기관의 시스템에서는 이들 암호 알고리즘이 필수적으로 요구된다.

이러한 세가지 요구 사항들을 만족시키며 암호화 통신 기능을 구현하기 위해 우리는 다음과 같은 방법으로 접근한다. 먼저, 기존의 통신 프로그램에 영향을 주지 않기 위해서 TCP/IP 프로토콜 이하의 수준을 수정한다. 통신 프로그램은 응용 수준(Application Layer)에 존재하므로 TCP/IP 이하 수준의 변경은 통신 프로그램에 영향을 주지 않는다. 또한, 기존의 통신 장비들이 사용하는 TCP/IP 프로토콜과의 호환성을 위해 표준 TCP/IP 에서 수정이 가능한 부분을 찾아 암호화 통신에 대한 추가적인 정보를 기록한다. 이를 위해, 기존의 IP 프로토콜의 헤더 중에서 추가 정보를 포함하는 options 필드[2]에 암호화 통신에 대한 정보를 추가한다. options 필드는 가변이므로, 이곳에 우리가 원하는 정보를 충분히 저장할 수 있다.

3. 암/복호화 모듈의 구조

[그림 1]은 암/복호화 모듈의 구조 도를 보여준다. 그림에서 점선으로 표시된 부분이 암/복호화 모듈이다. 패킷 암/복호화 모듈은 그림에서 보듯이 TCP/IP 프로토콜 처리부와 링크 프로토콜 처리부의 중간에 위치하여 링크 프로토콜 처리부에서 TCP/IP 프로토콜 처리부로 전달되거나 또는 반대 방향으로 전달되는 패킷에 대해 패킷 암/복호화를 수행한다. 패킷 암/복호화 모듈은 입력 처리부, 출력 처리부, 패킷 버퍼, 패킷 분석부, SDU(Service Data Unit) 암/복호화부, 패킷 헤더 계산부, 패킷 수정부로 구성되며, 구성 요소들의 기능은 다음과 같다.



[그림 1] 암/복호화 모듈의 구조

입력 처리부는 이전 프로토콜 처리부로부터의 패킷을 입력 받아 암/복호화를 위해 IP 패킷을 패킷 버퍼에 저장하고, 암/복호화를 위한 사전 처리를 수행한다.

패킷 버퍼는 암/복호화가 수행될 IP 패킷이 보관된다. 패킷 버퍼는 모든 처리부에 의해 공유된다. 패킷 버퍼의 데이터는 입력 처리부에 의해 기록되며, SDU 암/복호화부에 의해 SDU 데이터가 암/복호화 되며, 출력 처리부에 의해 다음 프로토콜 처리부로 출력된다.

출력 처리부는 패킷 버퍼에 저장된 IP 패킷을 다음 프로토콜 처리부로 출력하는 기능을 수행한다. 출력 처리부는 IP 패킷을 출력하기 전에 checksum 을 다시 계산하여, 암/복호화로 인해 IP 패킷의 checksum 오류가 발생하지 않도록 한다.

패킷 분석부는 암/복호화를 위해 입력된 패킷을 분석한다. 패킷 분석부는 패킷 버퍼로부터 패킷을 읽어서 패킷의 헤더를 분석하여 이 패킷이 암호화될 패킷인지 복호화될 패킷인지를 확인한다. 암호화되지 않은 패킷은 헤더의 옵션 부분에 별도의 표시가 되어 있지 않으므로 이런 패킷은 암호화를 수행하도록 결정하고, 헤더의 옵션 부분에 암호화 패킷에 대한 정보가 담긴 패킷의 경우에는 복호화를 수행하도록 결정하며, 패킷의 헤더를 분석하여 패킷 구조에 대한 정보를 작성한다. 또한, 암/복호화에 사용될 블록의 수, 나머지 부분의 크기, 나머지 부분을 블록의 크기에 맞추기 위해 추가되는 쓰레기 데이터의 크기를 계산하여, SDU 에 대한 정보를 작성한다. 이렇게 작성한 패킷 구조에 대한 정보는 패킷 헤더 계산부로 전달하며, SDU 에 대한 정보는 SDU 암/복호화부와 패킷 헤더 계산부로 전달한다. 이처럼, 패킷 분석부는 분석된 정보를 바탕으로 SDU 암/복호화부와 패킷 헤더 계산부에 암호화나 복호화 수행에 대한 정보를 전달한다.

SDU 암/복호화부는 패킷 분석부로부터 받은 SDU 에 대한 정보와 암/복호화 수행 결정에 따라 SDU 의 암/복호화를 수행한다. 먼저, 암/복호화 결과를 저장할 결과 저장용 메모리 공간을 할당한 후, 입력된 패킷의 SDU 에 대한 암/복호화를 수행하여 결과 저장용 메모리 공간에 저장한다. 암/복호화 결과는 별도의 결과 저장용 메모리 공간에 저장되므로, 원본 패킷의 내용은 수정 없이 보존된다. SDU 암/복호화부는 패킷에 적용되기 위해서 암/복호화 결과를 패킷 수정부로 보낸다.

패킷 헤더 계산부는 패킷 분석부로부터 받은 패킷 구조 정보와 SDU 정보를 기반으로 암/복호화로 인해 변경될 패킷 상태에 맞도록 패킷 헤더 정보를 수정한다. 블록 암호화를 수행하면 쓰레기 데이터의 추가로 인해 SDU 의 크기가 증가될 수 있고, 이런 경우 전체 패킷의 크기도 증가한다. 그러므로, 패킷 헤더 계산부는 패킷의 크기 정보를 추가된 쓰레기 데이터 크기만큼 증가시킨다. 그런데, 이 추가된 부분은 블록 크기를 맞추기 위해 추가된 쓰레기 데이터이므로, 나중에 복호화 후에는 이 쓰레기 데이터를 다시 감소시키기 위해 헤더 정보에 쓰레기 데이터에 대한 정보를 추가하여야 한다. 이를 위해 패킷 헤더 계산부는 패킷의 헤더 정보에 추가된 쓰레기 데이터의 크기에 대한 정보를 추가한다. 패킷 헤더 계산부는 헤더 정보를 수정한 후, 수정된 헤더 정보를 패킷 수정부로 전달한다.

패킷 수정부는 SDU 암/복호화부로부터 받은 SDU 의 암/복호화 결과 데이터와 패킷 헤더 계산부로부터 받은 패킷 헤더 정보를 이용하여 기존의 패킷을 수정

한다. 변경된 패킷 헤더는 크기가 변경되었으므로, 패킷 헤더의 크기만큼 패킷 버퍼의 크기를 다시 할당한 후, 여기에 수정된 헤더 정보를 기록한다. 또한, 암/복호화에 따라 SDU 의 크기 변경되었으므로, 변경된 SDU 의 크기에 맞도록 패킷 버퍼의 크기를 재 할당한 후, SDU 암/복호화부로부터 받은 SDU 결과를 기록한다. 패킷 수정부가 수정한 패킷은 패킷 버퍼를 통해 출력 처리부로 보내어진다.

4. 암/복호화 모듈의 동작 과정

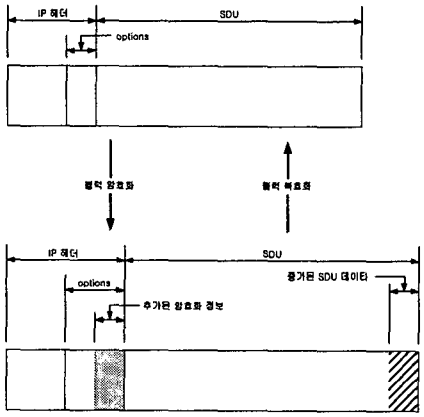
먼저, 암/복호화 모듈의 대략적인 동작 과정을 보면, 이전 프로토콜 처리부로부터 암/복호화 모듈로 입력된 IP 패킷은 입력 처리부에 의해 패킷 버퍼에 저장된다. 패킷 분석부에서 패킷 구조가 분석된 후, 헤더의 정보는 패킷 헤더 계산부로 보내지고, 패킷을 제외한 SDU 부분은 SDU 암/복호화부에서 암/복호화된 후, 패킷 수정부에서 헤더 정보와 암/복호화된 SDU 로 패킷을 수정하여 패킷 버퍼에 저장한다. IP 패킷에 대한 처리가 끝나면, 출력 처리부는 패킷 버퍼의 IP 패킷을 다음 프로토콜 처리부로 출력함으로써 암/복호화 모듈의 동작을 마친다.

패킷 암호화의 좀더 자세한 동작 과정은 다음과 같다. 암호화는 입력 처리부가 TCP/IP 처리부로부터 패킷을 전달 받아 패킷 버퍼에 저장하면서 시작된다. 패킷 분석부는 패킷 버퍼에 저장된 패킷을 분석하여, 패킷 헤더와 SDU 에 대한 정보를 만든다. 또한, SDU 데이터를 블록 암호 알고리즘으로 암호화하기위해 블록의 개수와 쓰레기 데이터의 크기 등의 정보를 계산한다. SDU 데이터를 블록 암호 알고리즘으로 암호화하면 결과가 기존의 SDU 데이터보다 커지게 되므로, 이를 바로 패킷 버퍼에 저장하면 할당된 패킷 버퍼의 공간을 초과하게 된다. 그래서 SDU 암/복호화부는 암호화된 결과를 임시로 저장하기위해 SDU 데이터 크기와 쓰레기 데이터를 합한 만큼의 결과 저장용 메모리를 할당한 후, 패킷 분석부가 작성한 정보를 이용하여 SDU 데이터를 암호화한다. 암호화에는 블록 암호 알고리즘이 사용되며, 암호화된 결과는 이전에 할당한 결과 저장용 메모리에 저장된다. 암호화가 끝나면, 패킷 수정부는 증가된 SDU 데이터와 암호화 정보를 추가하여 저장하기위해 패킷 버퍼의 크기를 증가시킨다. 패킷 헤더 계산부는 암호화로 인해 변경된 패킷의 상태를 반영하기 위해 패킷의 헤더를 수정한다. 패킷 헤더 계산부는 패킷의 SDU 가 암호화되었다는 것을 표시하기 위해 헤더의 options 필드에 암호화 정보를 추가한다. 헤더에 options 필드가 추가되었으므로, 이를 표시하기 위해 헤더의 header length 필드의 값을 증가시켜서, 헤더의 크기가 증가되었음을 기록한다. 또한 헤더의 증가와 SDU 데이터의 증가로 전체 패킷의 크기도 증가되었으므로, 패킷의 전체 크기를 나타내는 헤더의 total length 필드의 값도 증가된 패킷의 크기로 증가시킨다. 패킷 헤더의 수정이 끝나면, 패킷 수정부는 암호화된 SDU 데이터를 암호화 결과가 저장된 결과 저장용 메모리에서 패킷 버퍼로 복사하여 저장한 후, 암호화를 위해 할당된 결과 저장용 메모리를 제거한다. 출력 처리부는 변경된 패킷 데이터에 맞도록 checksum 을 다시 계산하여 헤더의 header checksum 필드에 기록한 후, 패킷 버퍼에 저장된 패킷을 링크 프로토콜 처리부로 전달한다.

다음으로, 암/복호화 모듈의 복호화 알고리즘을 살펴보자. 복호화는 입력 처리부가 링크 프로토콜 처리부로부터 패킷을 전달 받아 패킷 버퍼에 저장하면서 시작된다. 패킷 분석부는 패킷을 분석하여, 패킷 헤더와 SDU 에 대한 정보를 만든다. 또한 패킷 헤더를 분석하여, 쓰레기 데이터의 크기를 조사하고, SDU 를 블록 암호 알고리즘으로 복호화 하기위한 블록의 개수를 계산한다. SDU 데이터를 복호화하면 여기에는 암호화를 위해 추가된 쓰레기 데이터가 함께 존재하므로, 복호화된 데이터를 바로 사용할 수 없고, 쓰레기 데이터를 처리하는 과정을 거쳐야 한다. 이를 위해 SDU 암/복호화부는 복호화 된 결과를 임시로 저장하기 위하여 SDU 의 크기와 같은 결과 저장용 메모리를 할당한 후, 패킷 분석부가 작성한 정보를 이용하여 SDU 데이터를 복호화 한다. 복호화에는 블록 암호 알고리즘이 사용되며, 복호화 된 결과는 이전에 할당한 결과 저장용 메모리에 저장된다. 복호화가 끝나면, 패킷 수정부는 SDU 데이터에서 쓰레기 데이터가 제거된 크기와 헤더의 암호화 정보를 제거한 만큼 패킷 버퍼의 크기를 감소시킨다. 패킷 헤더 계산부는 복호화로 인해 변경된 패킷의 상태를 반영하기 위해 패킷의 헤더를 수정한다. 패킷 헤더 계산부는 패킷의 SDU 가 더 이상 암호화된 패킷이 아니라는 것을 표시하기 위해 헤더의 options 필드에서 암호화 정보를 삭제한다. 헤더에 options 필드가 감소되었으므로, 이를 표시하기 위해 헤더의 header length 필드의 값을 감소시켜서, 헤더의 크기가 감소되었음을 기록한다. 또한 헤더의 감소와 SDU 데이터의 감소로 전체 패킷의 크기도 감소되었으므로, 패킷의 전체 크기를 나타내는 헤더의 total length 필드의 값도 감소된 패킷의 크기만큼 감소시킨다. 패킷 헤더의 수정이 끝나면, 패킷 수정부는 복호화 된 SDU 데이터 중에서 쓰레기 데이터를 제외한 유효 데이터만을 복호화 결과가 저장된 결과 저장용 메모리에서 패킷 버퍼로 복사하여 저장한다. 이로써 패킷은 복호화를 마치고 암호화 이전의 패킷과 동일하게 된다. 패킷 수정부는 복호화를 위해 할당된 결과 저장용 메모리를 제거하고, 출력 처리부는 복호화된 패킷 데이터에 맞도록 checksum 을 다시 계산하여 헤더의 header checksum 필드에 기록한 후, 패킷 버퍼에 저장된 패킷을 TCP/IP 프로토콜 처리부로 전달함으로써, 패킷 복호화를 마친다.

[그림 2]는 블록 암/복호화에 따른 패킷 구조의 변화를 나타낸다. 위의 패킷은 암호화 되지 않은 패킷 구조를 나타내며, 아래 그림은 블록 암호 알고리즘을 이용하여 암호화된 패킷의 구조를 나타낸다. 패킷 암호화를 수행하면, 그림에서 회색으로 표시된 것처럼, 헤더의 options 필드에 암호화에 대한 정보가 추가된다. 또한, 암호화의 결과로 SDU 의 크기가 블록의 N 배에 해당하는 크기로 증가하는데, 그림에서 사선으로 표시된 부분이 증가된 SDU 부분이다. 반대로, 암호화

된 패킷에 대하여 블록 암호 알고리즘을 이용하여 복호화를 수행하면, 암호화된 패킷의 헤더 부분에서 options 필드의 암호화 정보가 삭제되고, SDU 데이터 중에서 쓰레기 데이터 부분이 삭제되어, 암호화 이전의 패킷 구조를 가지게 된다. 암/복호화는 패킷의 크기를 변경시킬 뿐만 아니라 패킷의 헤더 정보를 변경한다. 헤더 정보는 패킷의 구조에 대한 정보를 포함하므로, 암/복호화에 따라 변경되는 패킷의 구조를 표시하기 위해 패킷의 헤더 정보는 암/복호화 후에 내용이 변경된다.



[그림 2] 암/복호화에 따른 패킷의 변화

5. 암/복호화 모듈의 구현 사례 : SEED 와 Linux

설계된 암/복호화 모듈을 리눅스에 적용하여 테스트 하였다. 리눅스는 모든 소스가 공개되어 있고, 관련 자료가 풍부하여, IP 패킷을 직접 수정해야 하는 이번 작업에 적합하다는 판단에서였다. 리눅스에서 TCP/IP 프로토콜에 대한 코드는 커널에 포함되어 있으며, 내부적으로 소켓 버퍼를 이용한다. 이는 커널 소스에 sk_buff 구조체로 정의되어 있다. TCP/IP 처리 루틴들은 이 sk_buff 를 이용하여 패킷 데이터를 처리하거나, 전달한다[3]. 커널 소스는 버전 2.2.16 을 사용하였고, IP 패킷을 수정하고, 암/복호화 모듈을 리눅스에 적용하기위해서 커널의 TCP/IP 관련 소스를 수정하여, 암/복호화 루틴을 추가하였다. 블록 암/복호화 알고리즘으로는 한국정보보호진흥원(KISA)에서 개발한 128 비트 대칭형 블록 암호 알고리즘 표준(SEED)을 이용하였으며, 리눅스에 적용하기 위해서 SEED 알고리즘 소스를 리눅스 커널에 추가하여 커널과 같이 컴파일 하였다.

블록 암/복호화는 IP 패킷의 크기를 변경시킬 수 있으므로, 리눅스의 TCP/IP 처리 루틴에서 전송을 위해 이더넷의 MTU(Maximum Transfer Unit) 크기로 이미 분할한(fragmented) 패킷을 암호화하면 패킷이 MTU 크기 이상으로 증가해서 패킷을 다시 분할해야 하는 경우가 발생할 수 있다. 그래서 구현의 간단화를 위해 적용 범위를 end-to-end 환경으로 제한하여, 암호화는 패킷을 분할하기 이전에 수행하고, 복호화는 분할된 패킷이 하나로 합쳐진(defragmented) 이후에 수행한다 [4].

4 장의 암호화 동작 과정을 리눅스에 적용하면, 대략적으로 아래와 같다. 복호화 과정은 아래 내용을 4 장의 복호화 동작 과정과 비교하여 동일하게 적용하면 된다.

1 버퍼 할당 및 sk_buff 복사
2 추가될 쓰레기 데이터 크기 계산
3 sk_buff 크기 늘리기
4 sk_buff.ip_hdr 로 IP 헤더 수정
5 버퍼 데이터 암호화 및 sk_buff 에 저장
6 ip_send_check()로 IP checksum 다시 계산

IP 헤더 수정 부분에서는 패킷에 암호화 정보를 추가하기 위하여, IP 헤더의 options 필드 끝부분에 SEED 암호화에 대한 정보를 추가한다. 추가되는 정보는 SEED 표시자(SEED 로 암호화된 패킷임을 표시), 정보 크기(암호화 정보를 표시하기 위해 options 필드에 저장된 데이터의 크기), 쓰레기 크기(블록 크기를 맞추기 위해 추가된 쓰레기 데이터의 크기), IPOPT_NOOP(빈 공간임을 표시-IP 헤더의 크기는 4 의 배수를 유지해야 하기 때문에 필요에 따라 추가), IPOPT_END(options 필드의 끝임을 표시)이다.

6. 결론

중간 노드들에서 통신 데이터를 불법으로 획득하는 스니핑에 대한 대안으로 암호화 통신에 대한 요구가 증가하고 있다. 이를 위해 본 논문에서는 암호화 통신을 위해 블록 암호 알고리즘을 사용한 패킷 암/복호화 모듈을 제시한다. 이 모듈은 기존의 네트워크와 응용 프로그램에 영향을 주지 않고 암호화 통신 기능을 제공하기위해 IP 패킷의 헤더 정보를 변경하는 방법을 사용한다.

이러한 방법을 이용하면 암호화 통신에 다양한 크기의 블록 암호 알고리즘을 사용할 수 있으므로, 암/복호화 속도와 안정성 면에서 보다 나은 성능을 얻을 수 있다. 블록 암호 알고리즘은 보다 긴 암/복호 키를 사용하므로 암호화된 데이터에 대한 신뢰성도 보다 높아지는 장점을 가진다. 이와 더불어, 암/복호화를 위해 기존의 스트림 암호 알고리즘 뿐만이 아니라 다양한 블록 암호 알고리즘을 사용할 수 있으므로, 암호 알고리즘에 대한 선택의 폭을 넓혀준다.

참고문헌

[1] 한국정보보호진흥원, "128 비트 블록 암호 알고리즘(SEED) 개발 및 분석 보고서", 1999
[2] W. Richard Stevens, "TCP/IP Illustrated, Volume 1 : The Protocols", Addison-Wesley, 1994
[3] David A Rusling, "The Linux Kernel", http://kldp.org/Translations/html/The_Linux_Kernel-KLDP/The_Linux_Kernel-KLDP.html
[4] Glenn Herrin, "Linux IP Networking : A Guide to the Implementation and Modification of the Linux Protocol Stack", http://kernelkorea.org/html/linuxkernel/doc/linux-net.pdf