情報保護學會論文誌
第16卷 第5號, 2006. 10

IPsec System에서 IKEv2 프로토콜 엔진의 구현 및 성능 평가

김 성 찬,[1†] 천 준 호, 전 문 석[2‡]
[1](주)유코레일, [2]숭실대학교

An Implementation and Performance Evaluation of IPsec System engaged IKEv2 Protocol Engine

Sung-Chan Kim,[1†] Jun-Ho Chun, Moon-Seog Jun[2‡]
[1]EUKORAIL Co. Ltd, [2]Soong-Sil University

요 약

기존의 보안 시스템에서 키 교환에 사용되어 왔던 인터넷 키 교환 프로토콜은 확장성과 속도, 효율성 그리고 안정성에 문제가 있다는 지적을 받아 왔다. 본 연구에서는 이러한 문제점을 해결하고자 했으며, 새로 구현한 인터넷 키 교환 프로토콜을 IPsec에 연동되는 테스트베드에 탑재하여 성능평가를 하였다. 네트워크가 점점 확장되어 가는 상황에서 기존의 인터넷 키 교환 프로토콜은 네트워크의 확장성에 한계가 있기 때문에 RFC에서 제안하는 표준에 따라서 구현 하였으며 키 교환 및 인증 과정에서 지적된 복잡성과 속도 문제를 해결 하였다. 복잡한 메시지 교환 단계를 줄여서 속도를 향상시켰고, 재협상 시 기존 상태 값들을 재활용함으로써 효율성을 증가 시켰다.

ABSTRACT

The current Internet Key Exchange protocol(IKE) which has been used for key exchange of security system was pointed out the faults of scalability, speed, efficiency and stability. In this research, we tried to resolve those faults, and implemented the newly designed IKEv2 protocol in the IPsec test bed system. In the trend of network expansion, the current Internet Key Exchange protocol has a limitation of network scalability, so we implemented the new Internet Key Exchange protocol as a recommendation of RFC proposal, so as to resolve the fault of the key exchange complexity and the speed of authentication process. We improved the key exchange speed as a result of simplification of complex key exchange phase, and increased efficiency with using the preexistence state value in negotiation phase.

Keywords : *IPsec, IKE, IKEv2, Security*

Ⅰ. 서 론

현재 정보시스템간의 데이터 처리 및 전송되는 정보는 외부에 노출된 공중망을 비롯한 인터넷 통신망의 급속한 확장에 따라 디지털화 대용량화 하고 있어 데이터에 대한 적절한 보호 조치가 없으면 송수신, 처리 과정이나 기억장치에 보관된 상태에서 불법유출, 삭제 및 수정 등의 위험에 노출되기 쉽다. 이러한 불법적인 사고로 인하여 개인 사생활 침해뿐만 아니라 막대한 경제적 손실을 당할 우려가 있어 정보보

접수일: 2006년 6월 3일; 채택일: 2006년 8월 21일
† 주저자, viper72@chol.com
‡ 교신저자, mjun@computing.ssu.ac.kr

호에 대한 관심은 점점 고조되고 있다. IPsec 프로토콜이란 보안이 취약한 인터넷에서 전달되는 IP 패킷을 대상으로 패킷이 전송 중에 데이터가 변하지 않았음을 보장하고, 발신자의 사용 인증 및 데이터 암호화를 통해 정보가 노출되지 않았음을 보장하는 프로토콜로써 보안 서비스를 제공해 주는 인터넷 보안 메커니즘의 하나이다[1]. IPsec 프로토콜에서 송수신자 양측은 사전에 데이터를 암호화하고 암호화된 키를 복호화 하기 위한 공유키를 안전하게 공유하고 있어야 한다. 이러한 키 공유 작업을 안전하게 자동으로 해주는 프로토콜이 키 관리 프로토콜이며, 이것을 IETF(Internet Engineering Task Force)에서는 인터넷 키 교환(IKE) 프로토콜로서 RFC 2407, 2408, 2409 문서로 표준화 하였다[2][3] 하지만 기존의 인터넷 키 교환(IKE) 프로토콜은 기능이 복잡하고 표준 규격의 기술도 난해하여 이를 구현한 제품간에 상호 연동이 힘들었다. 본 연구에서는 2005년 12월에 제출된 RFC4306에서 제안하는 향상된 인터넷 키 교환(IKEv2)프로토콜을 IPsec 프로토콜과 연동되도록 구현하여 실험실 내의 테스트 베드 환경에서 패킷을 분석하고 기대되는 바와 같이 동작하고 속도가 향상되었는지를 확인하였다. 본 논문의 구성은 다음과 같다. Ⅱ장에서는 본 연구의 기초가 되는 관련연구에 대하여 기술하고, Ⅲ장에서는 본 논문에서 구현한 IKEv2 프로토콜과 IPsec 테스트 베드 시스템에 대하여 기술한다. Ⅳ장에서는 구현된 시스템에 대한 실험 결과, 산출물로서 암호화된 패킷 메시지 분석과 키 교환 속도를 비교한 평가를 한 후, Ⅴ장에서 결론을 맺는다.

Ⅱ. 관련연구

2.1 IPsec 프로토콜

IPsec은 IP Security의 약자로 IP 패킷에 대한 보안을 제공하며 IP 패킷의 인증 및 무결성, 기밀성 서비스를 제공한다. IPsec은 AH(Authentication Header)와 ESP(Encapsulating Security Protocol), IKE(Internet Key Exchange)의 3가지 프로토콜로 구성된다. 이중, AH나 ESP 프로토콜은 대칭키를 기반으로 한 암호 알고리즘을 사용하고 있는데, 이 프로토콜들을 위한 키의 생성, 유지, 갱신 및 분배에 대한 프로토콜은 키 교환 메시지를 이용해 키를 생성 및 교환하게 된다. 다음에 IETF에서 표준화된 IPsec 프로토콜과 그 동작 모드 및 키 관리 프로토콜에 대해 간략히 기술한다.

2.2 AH 프로토콜

AH 프로토콜은 인증 알고리즘을 이용하여 패킷 무결성 서비스와 데이터 근원 인증 서비스를 제공한다. 그림 1은 AH 헤더 포맷을 보여 준다[4].

2.3 ESP 프로토콜

인증 헤더(AH)의 경우와 마찬가지로 ESP (캡슐화 보안 페이로드) 프로토콜은 IP 프로토콜의 보안성을 향상시키기 위해 설계 되었다. ESP는 AH가 제공하는 서비스에 데이터 기밀성과 제한된 트래픽 흐름 기밀성의 두 가지 서비스를 추가로 제공한다. 그림 2는 ESP 프로토콜의 헤더 포맷을 보이고 있다[5].

2.4 프로토콜 동작 모드

AH와 ESP 헤더의 위치는 프로토콜의 동작 모드에 따라 달라지는데, 각 프로토콜의 동작 모드에는 전송 모드와 터널 모드가 있다. 먼저, AH 프로토콜의 전송 모드의 동작을 보면, 그림 3과 같이 IP 헤더의 뒤, 전송 계층 프로토콜 헤더의 앞, 또는 다른

IP 헤더	Protocol = 51 (AH)	20
	Next Header	1
	Payload Length	1
	RSVD	2
	Security Parameter Index (SPI)	4
	Sequence Number	4
	Authentication Data	m (가변 길이=12)
	Data	n

그림 1. IPsec AH 프로토콜 형식

IP 헤더	Protocol = 50 (ESP)	20
ESP 헤더	Security Parameter Index (SPI)	4
	Sequence Number	4
Payload	암호화된 Data	n
ESP Trailer	Padding	0 ~ 255
	Pad Length	1
	Next Header	1
ESP Auth	Authenticator (HMAC)	n

그림 2. IPsec ESP 프로토콜 형식

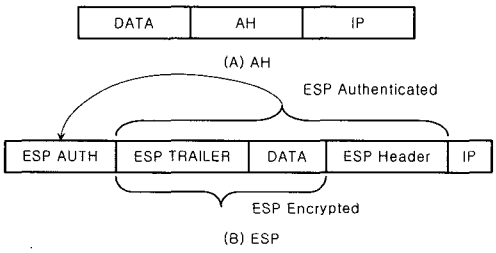

그림 3. 전송모드의 AH, ESP 헤더 위치

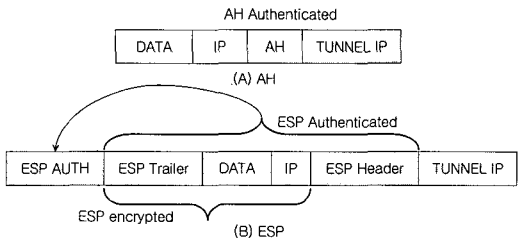

그림 4. 터널모드의 AH, ESP 헤더 위치

IPsec 프로토콜 헤더들의 앞에 삽입된다.

전송 모드로 동작되는 ESP 프로토콜의 경우도 이와 유사하게 ESP 헤더가 AH헤더 위치에 삽입된다. 물론, ESP프로토콜의 경우 상위 계층 데이터의 뒤에 ESP 트레일러가 뒤따른다. 또한, AH와 ESP 프로토콜은 터널 모드로 동작 할 수 있다. 그림 4는 AH 프로토콜이 터널 모드로 동작될 때 헤더의 위치를 보여 준다. AH 헤더는 원래의 IP 헤더 앞에 삽입되며, 새로운 IP 헤더가 AH 헤더 앞에 삽입된다. ESP 헤더도 동일한 위치에 삽입된다.

2.5 IKE 프로토콜

IKE는 보안연계(SA: Security Association)의 수립을 위하여 인증된 키 자료를 보호된 방식으로 협상하고 제공하는 프로토콜이다. IKE는 3개의 서로 다른 프로토콜의 관련 부분을 결합한 하이브리드 프로토콜로서, ISAKMP(Internet Security Association and Key Management Protocol), Oakley Key Determination Protocol과 SKEME 프로토콜로 이루어져 있다. ISAKMP 프로토콜에서는 프레임워크, 메시지 포맷 및 Phase (단계) 개념을 채용해 왔고, Oakley 프로토콜에서는 두 가지 키 교환 모드, 그리고 SKENE 프로토콜에서는 공개키 암호방식을 가져왔다. IKE는 ISAKMP 협상의 단계에서 동작하는 여러 교환 모드를 제공하며, 1단계 교환에서 개시자와 응답자간에 ISAKMP SA를 수립하고, 2단계에서는 AH, ESP와 같은 다른 보안 서비스를 위한 SA를 수립하는데 이용된다. 그림 5에서 IKE의 동작 절차를 나타내었다[6][7].

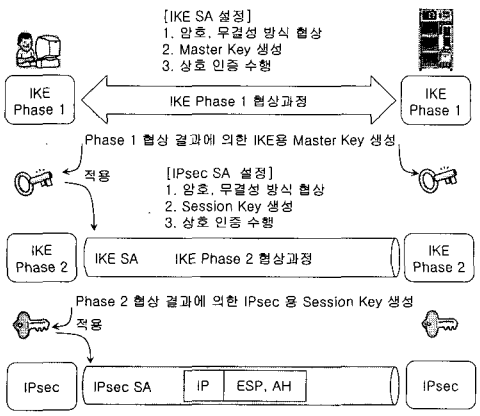

그림 5. IKE 동작 절차

1단계 에서는 안전하고 인증된 통신 채널을 생성하고, 인증된 키 교환을 수행하고, 메인(Main) 모드 또는 어그레시브 (Aggressive) 모드 중 한 모드로 동작함으로써 안전한 IKE SA를 수립한다[8]. 메시지 교환은 모두 ISAKMP 메시지를 이용하며, ISAKMP 헤더 포맷은 그림 6과 같다.

Phase I에서 메인 모드의 경우 메시지 교환 절차는 그림 7과 같다. 그림에서 보는 것과 같이 모두 여섯 개의 메시지 교환으로 이루어진다. Phase I에 의해 IKE SA가 수립되면, Phase II에서는 실제로 보안 통신용 사용자 SA가 생성되는데, 그 중 하나가 IPsec SA이다. 즉, DOI(Domain of Interpretation)에 다른 보안 프로토콜의 SA 생성이 가능하다[9][13]. 이를 위한 Phase II의 퀵 (Quick) 모드 메시지 교환은 그림 7과 같다.

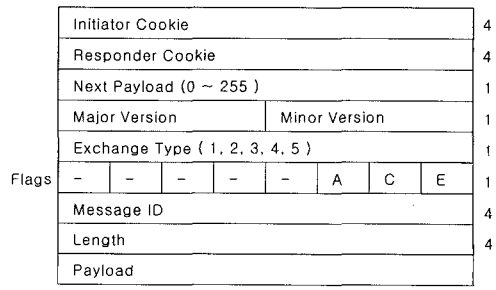

그림 6. ISAKMP 헤더 형식

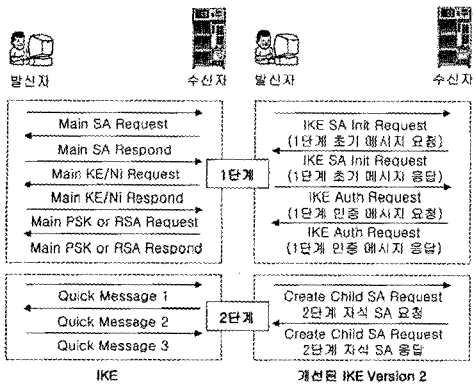

그림 7. IKE와 IKEv2 Phase I II 메시지 교환

2.6 향상된 IKE 프로토콜 IKE Version 2

RFC 2409로 표준화 되어 있는 IKE 프로토콜은 기능이 복잡하고 표준 규격의 기술(Description)도 난해하여 이들을 구현 한 서로 다른 기종 제품 간에 상호 연동성이 아직까지도 확보되지 않고 있다. 이러한 배경에서 IETF IPsec 워킹 그룹은 2001년 8월에 새로운 키 관리 프로토콜을 개발키로 하였고 그동안 두 프로토콜 즉, IKE Version 2와 JFK(Just Fast Keying)프로토콜이 경합되어 왔으나 최근 IKEv2로 표준화를 조율하여 2005년 12월 RFC4306 문서가 제출 되었다[7][10]. IKEv2는 기존의 IKE 프로토콜의 개념을 그대로 계승하면서 기능을 대폭 축약하였다. IKEv2는 새로운 키 관리 프로토콜이 만족시켜야 하는 요구사항을 충족하도록 설계되었다[11][12]. IKEv2는 정상 상태에서 4개의 메시지 교환에 의해 통신 쌍방이 인증된 보안 채널을 수립할 수 있다. IKEv2는 기존 IKE Phase I/II 분리 개념을 계승하여 Phase I에서 수립된 IKE SA를 후속 IPsec SA들의 수립과 관리 시 제어 채널로 활용할 수 있게 하였다[13][15]. 인증 방식 에서는 IKEv2는 공개키 기반의 인증서에 의한 인증을 기본으로 하지만, 사전 공유키 사용도 지원한다.[14][18]

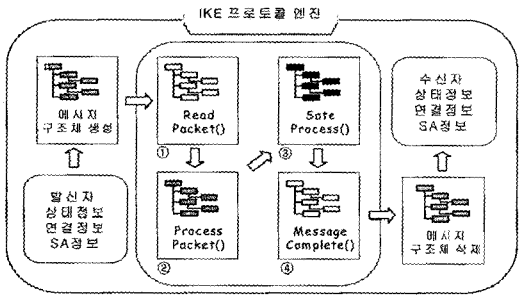

그림 8. IKEv2 프로토콜 엔진

III. IKEv2 프로토콜 엔진 과 IPsec 연동 테스트 베드 구현

본 연구에서는 새롭게 키 교환 프로토콜의 표준으로 제안된 IKEv2 프로토콜을 IPsec 프로토콜과 연동되도록 구현하여 그 성능을 기존의 IKE 프로토콜과 비교해서 평가하는 것을 목표로 삼았다. 새롭게 개선된 IKEv2 프로토콜 엔진을 IPsec 공개 소프트웨어[16][17]에 탑재하여 실험실 내에 IPsec VPN 테스트 베드를 구현하고 IPsec 게이트웨이 간에 보안 통신 수립 시 교환되는 패킷을 분석하여 암호화의 여부를 분석하고 속도를 측정하여 평가 하였다.

3.1 IKEv2 프로토콜 엔진의 구현

본 논문에서 구현한 IKEv2 프로토콜 엔진은 크게 4개의 패킷 처리 함수로 구성되어 있다. 그림 8은 IKE 메시지를 처리하는 IKE 프로토콜 엔진 모듈을 보여주고 있는데 4단계[19]를 거쳐 패킷을 처리한다. 먼저 상대방 IKE 소켓에서 메시지를 읽어오는 read_packet() 함수, 두 번째는 읽어 온 메시지를 파싱하는 process_packet() 함수, 세 번째는 받은 페이로드를 하나하나 처리하고 응답메시지를 생성하여 상태 정보 구조체에 반영하는 state_process()

그림 9. 주요 자료구조 및 DATA FLOW

함수, 마지막으로 응답 메시지를 해당 소켓으로 보내고, 관련된 이벤트를 스케줄 하는 message_complete() 함수 이다.

그림 9는 IKEv2 프로토콜 엔진의 자료 흐름에 사용되는 구조체들의 흐름을 나타낸 것이며, IKEv2 프로토콜 엔진의 핵심 구성 요소이다. 그림 9의 ①번 상태 종류 구조체에 사용되는 상태 종류 변수는 IKE 메시지 교환 단계에 대한 상태를 명시적으로 지정하는 변수이며, ②번 메시지 구조체는 해당되는 메시지 교환 단계에 대한 정보를 가지고 있고, IKEv2 모듈이 호출 될 때 생성되어 끝날 때 사라지는 임시 변수이다. 메시지 교환의 연결 정보라든지, 받은 메시지가 몇 번째 교환 상태인지, 보낼 메시지 데이터가 무엇인지 등의 정보를 담고 있으며, 메시지 구조체의 상태 종류 필드는 상태 절차 설명 구조체의 현재 상태 필드와 연결 되어 있다. ③번 상태 절차 설명 구조체는 각 상태별로 처리해야 하는 일련의 절차상의 정보를 가지고 있다. 이 상태 종류는 몇 번째 교환에 해당하고, 다음 상태는 어떤 것이며, 받는 페이로드와 보낼 페이로드들이 어떤 것이 되어야 한다고 규정되어 있다. 상태 처리 함수는 실제 페이로드들의 값을 처리하여 SA를 협상하는 함수로써 각 상태 종류마다 별도로 구현되어 있다. SA 페이로드를 분석하여 해당 알고리즘을 설정하고, 공개키나 비밀키등을 계산하며, 그 값을 통해 암호화와 인증을 하는 실제 처리를 담당한다. 그리고 협상된 모든 정보를 상태 정보 구조체라는 자료구조에 반영한다. 상태 정보 구조체는 전역변수로 잡혀 있는 구조체로서 연결정보와 SA협상에 관련된 모든 정보를 담고 있다. 이 상태 정보 구조체를 통해 각 메시지 교환 단계에서 SA 협상을 하게 되며, 그 정보를 유지하고 있다. IKE 메시지 교환을 통해 완벽하게 협상된 이 구조체의 SA정보는 커널에 있는 보안 데이터베이스에 반영하도록 하였다.

그림 10은 그림 9의 ①번 상태 종류 구조체에 사용되는 상태 종류 변수를 나타내고 있다. 이 변수는 메시지 교환 단계를 나타내며, 메시지 구조체의 상태 종류 필드와 상태 절차 설명 구조체 배열의 인덱스로 활용된다. 즉 상태 종류 변수는 각 처리 단계를 구분하는 식별자로써 IKE 메시지가 들어 왔을 때 몇 번째 교환 절차인지를 명시한다. 1단계 초기 교환 시 송신자가 보낸 요청 메시지를 수신자측이 받은 상태는 STATE_PHASE1_R0 가 된다. 또한 수신자가 이에 대한 응답 메시지를 보내어 송신자가 그 메시지를 받은 상태는 STATE_PHASE1_I1이 된다. 이러한 방식으로 그림 10과 같이 IKE 키 교환 단계 마다 상태 종류를 명시하였고, 상태마다 적합한 처리 절차와 연결시킴으로써 해당 교환 단계에 맞는 메시지 처리가 이루어지도록 구현하였다.

IKEv2 프로토콜 모듈은 먼저 각 메시지 교환의 정보를 담는 구조체인 그림 11과 같은 메시지 구조체를 인자로 받는다. 먼저 통신하는 두 호스트간의 연결정보 필드와 받은 메시지 스트림에서 헤더와 페이로드별로 파싱하여 별도의 필드로 관리한다. 발신자로부터 IKE 메시지가 전송되면 수신자는 먼저 연결 정보를 채우고, 그 메시지를 받아서 메시지 구조체에 헤더와 페이로드별로 파싱하여 필드를 채운다. 1단계 초기 메시지 교환 단계에서는 HDR와 SA, KE, Ni(발신자용 Nonce 페이로드)를 발신자로부터 받게 된다. 수신자는 이 정보를 가지고 SA를 협상하기 위한 처리를 하고, 응답 메시지인 HDR와 SA, KE, Nr(응답자용 Nonce 페이로드)을 생성하여 메시지 구조체의 보낼 메시지 필드에 채운다. 그리고 이 단계에서 필요한 작업이 모두 끝나면, 메시지 구조체에 반영하고 나서 메시지 구조체는 삭제되고, 상태 정보 구조체의 보낼 메시지 필드의 내용을 발신자에게 보내면서 하나의 교환 단계가 마무리 된다.

IKE 메시지가 들어 왔을 때 헤더를 분석 해 보면

```
enum state_kind {

STATE_PHASE1_R0, // 1단계 초기 메시지 교환 (응답자 측)
STATE_PHASE1_I1, // 1단계 초기 메시지 교환 (발신자 측)
STATE_PHASE1_R1, // 1단계 인증 메시지 교환 (응답자 측)
STATE_PHASE1_I2, // 1단계 인증 메시지 교환 (발신자 측)

STATE_PHASE1_R2, // 1단계 SA 설립 완료 상태 (응답자 측)
STATE_PHASE1_I3, // 1단계 SA 설립 완료 상태 (발신자 측)

STATE_PHASE2_R0, // 2단계 자식 SA 생성 요청 메시지 교환 (응답자 측)
STATE_PHASE2_I1, // 2단계 자식 SA 생성 응답 메시지 교환 (발신자 측)

STATE_PHASE2_R1, // 2단계 SA 설립 완료 상태 (응답자 측)
STATE_PHASE2_I2, // 2단계 SA 설립 완료 상태 (발신자 측)
};
```

그림 10. 상태 종류 변수

```
Sturct msg_digst {
Const struct iface *iface;                    // 인터페이스
  ip_address sender;                          // 발신자주소
  u_int16_t sender_port;                      // 발신자포트
  pb_stream packet_pbs;                       // 받은메시지(스트림)

  struct isakmp_hdr hdr;                      // 받은 메시지의 헤더(헤더구조체)
  struct payload_digest  digest[PAYLIMIT]; // 받은 메시지의 페이로드 배열

  enum state-kind from_state;                 // 상태 종류

Pb_stream reply;                              // 보낼 메시지(스트림 헤더 포함)
};
```

그림 11. 메시지 구조체 필드

```
Sturct state microcode {
Enum state_kind state;                  // 현재 상태를 지정
      enum state_kind next_state;       // 다음 상태를 지정
      lset_t flags;                     // 인증 및 암호화 옵션 플래그
      lset_t req_payloads;              // 필수 페이로드 비트셋
      lset_t opt_payloads;              // 옵션 페이로드 비트셋
U_int8_t
      first_out_payload;                // 응답용 메시지의 최초 페이로드
      enum event_type timeout_event // 상태와 연관된 이벤트(재전송,만기)
      state_transition_processor;       // 상태 처리함수의 포인터
};
```

그림 12. 상태 절차 구조체

그 메시지가 지금 어떤 상태 종류인지 알 수가 있고, 상태 종류를 인덱스 삼아 해당 상태 절차 설명 구조체를 찾아 간다. 그리고 상태 절차 설명 구조체의 상태 처리 함수의 포인터 필드를 참고하여 메시지 교환 처리를 한다. 그림 12는 상태 절차 설명 구조체를 나타낸다. 상태 절차 설명 구조체는 현재 상태 종류와 다음에 처리되야 할 상태 종류, 이번 단계에 들어와야 되는 필수 페이로드 목록과 옵션 페이로드 목록, 관련 이벤트 종류 등의 처리 절차상의 중요한 정보를 담고 있다. 또한 키 생성, SA 설립 등의 실질적인 처리를 하는 상태 처리 함수의 포인터를 가지고 있어서 상태에 맞는 적절한 상태 처리 함수를 이어주는 역할을 한다. 그리고 스트림 형태로 전달되는 IKE 메시지를 처리하는 그림 8의 함수들을 자세히 설명하면 다음과 같다.

그림 8의 IKE 프로토콜 엔진은 각 교환 단계의 정보를 담은 메시지를 받은 소켓 정보로부터 발신자 IP 주소와 포트 정보의 연결정보를 얻어 와서 메시지 구조체의 연결정보 필드를 채운다. 그 후에 UDP 소켓에서 읽은 IKE 메시지를 적절한 크기로 잘라서 메시지 구조체의 받은 메시지 스트림 필드로 채워 넣는다. 따라서 그림 13의 read_packet 함수는 UDP 소켓으로 받은 메시지를 적절히 잘라서 메시지 구조체의 받은 메시지 필드에 스트림 형태로 받아들이는 함수이다. 연결 정보 필드는 응답 메시지를 보낼 때 해당 주소로써 관리가 되고, 받은 메시지 스트림 정보는 그림 14의 process_packet 함수에서 파싱되어 각 각의 페이로드로 분해된다. 파싱된 헤더에는 현재 어떤 교환 절차인가에 대한 헤더 교환 종류 정보를 가지고 있으며, 그림 15의 State Process 함수에서 헤더 교환 종류를 보고 1단계인지 2단계인지, 최초 메시지인지 두 번째 메시지인지, 수신자인지 발신자 인지 등을 파악하여 해당하는 상태 종류를 찾아낸다. 또한 스트림으로 되어 있는 데이터가 암호화 되어 있을 시에는 복호화를 한 후에 각 페이로드와 필드의 특성에 맞게 파싱하여 페이로드 구조체 배열로 관리 유지하고 상태 종류를 찾아내어 현 단계에 맞는 상태 처리 함수를 호출 할 수 있다. 그림 15 같이 상태 처리 함수는 process_packet 함

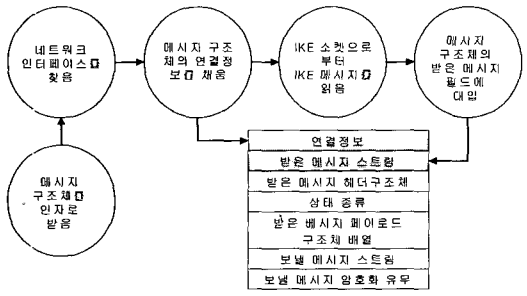

그림 13. Read Packet() 처리

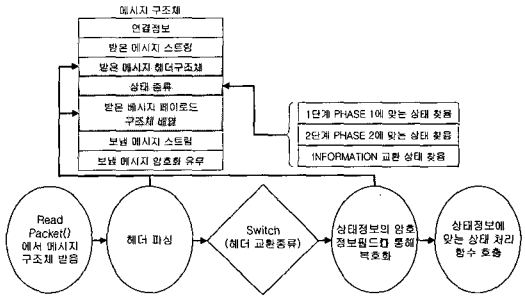

그림 14. Process Packet() 처리

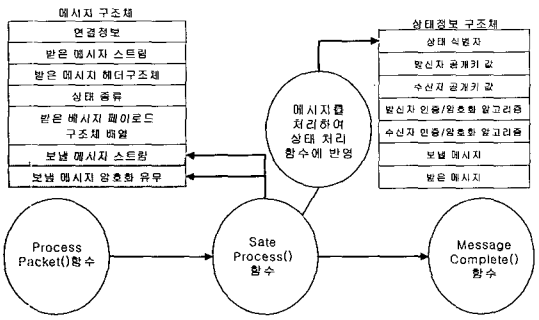

그림 15. State Process() 처리

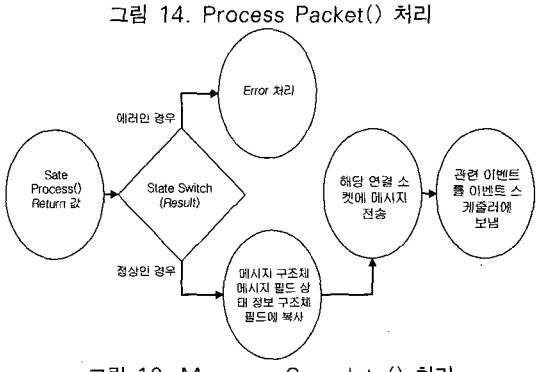

그림 16. Message Complete() 처리

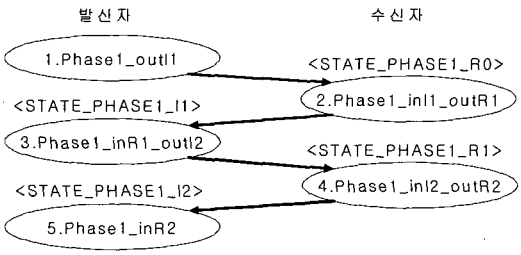

그림 17. 1단계 상태 처리

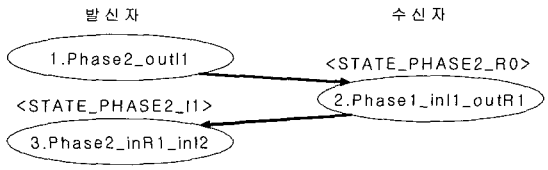

그림 18. 2단계 상태 처리

수를 통해 메시지 구조체를 인자로 받아서 페이로드를 분석, 계산하여 키 값을 산출해 내고 전역 구조체인 상태 정보 구조체에 반영한다. 또한 보낼 메시지를 생성하여 메시지 구조체의 보낼 메시지 스트림 필드에 보관 후 그림 16의 Message Complete()함수로 보내 모든 처리가 정상적으로 끝났을 경우 해당 연결 소켓에 메시지를 보내어 메시지 교환을 마무리하고 비정상일 경우 에러를 리턴 한다.

그림 17과 18은 1,2 단계의 상태 처리 함수를 나타낸다. 상태처리 함수에서의 메시지 교환 1단계는 2쌍의 메시지 교환으로 이루어져 있고 송신자와 수신자 별로 상태 종류가 구분된다. 송신자와 수신자 간의 최초 연결 설립을 요청하는 경우에는 별도의 상태 종류가 필요 없고 상태 처리 함수만 존재 하며 최초 수신시부터 상태가 적용된다. 그림 17의 1번 상태에서 메시지 교환 1단계의 발신자측 메시지를 받으면 발신자측의 상태 정보 구조체를 새로 생성하게 된다. 2번 상태에서는 수신자측의 상태 정보 구조체를 패킷 프로세스를 통해 파싱되어 받은 페이로드들을 하나씩 처리하여 상태 정보에 반영하고 응답용 페이로드를 만들어서 메시지 구조체의 보낼 메시지 스트림에 채우게 된다. 3번 상태에서는 해당 1단계 상태 정보를 찾아내어 암호화를 초기화 하고 인증 처리를 하게 된다. 4번 상태에서는 초기화된 메시지를 식별하여 인증하고 1단계 수신자측의 SA 설립을 완료한다. 마지막 5단계에서는 발신자측의 메시지를 식별하여 인증하고 1단계 발신자측 SA 설립을 완료한다.

그림 18의 2단계 상태 처리에서는, 1번 상태에서 2단계를 위한 발신자측의 상태 정보를 새로 생성 후에 기존의 1단계 상태 정보의 값을 복사하고 페이로드를 생성한다. 상태정보의 암호화 정보를 통해 Hash 이후의 페이로드를 암호화 한 후 메시지 처리 완료 함수로 보낸다. 2번째 상태에서는 2단계를 위한 수신자측 상태 정보를 새로 생성한 후에 기존의 1단계 상태 정보의 값을 복사하고 파싱된 페이로드들을 하나씩 처리하여 상태 정보에 반영한다. 1단계에서 사용된 상태정보를 이어서 사용함으로써 상태 전이 횟수를 줄이게 되고 마지막 단계에서 커널에게 2단계 SA가 설정되었음을 알린다. 그리고 메시지 처리 함수에서는 받은 메시지의 정상 유무를 판단하여 정상일 경우 복호화 하여 해당 이벤트 처리기에게 보내고 비정상일 경우 에러 메시지를 리턴 하게 한다. 1단계는 2쌍의 메시지 교환, 2단계는 1쌍의 메시지 교환으로 이루어져 있고, 발신자와 수신자 별로 상태 종류가 구분된다. 본 논문에서는 줄어든 절차와 통합된 페이로드로 동작 하도록 구현 하였다.

3.2 IPsec 연동 테스트 베드 구현

본 논문에서 구현한 IKEv2 프로토콜 엔진의 성능 평가를 위해 IPsec 공개 소프트웨어를 사용하여 구축한 IPsec 프로토콜에 IKEv2 프로토콜을 탑재하여 키 교환 및 SA 성립 후에 VPN 인증 터널을 구현하는데 소요되는 시간을 측정하여 그 성능을 기존의 키 교환 프로토콜과 비교 하였다. 테스트 베드의 구성은 그림 19와 같다.

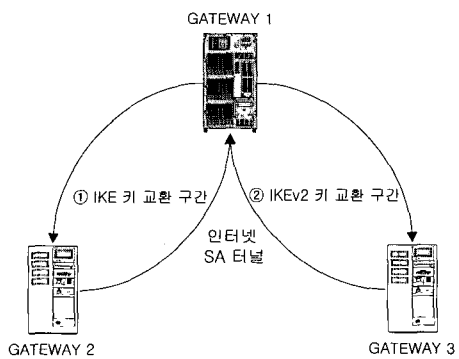

그림 19. 테스트 베드 구성도

본 연구에서 구현한 IKEv2 프로토콜과의 비교 평가를 위해 기존의 IKE 프로토콜 엔진과 IKEv2 프로토콜 엔진이 탑재된 VPN 게이트웨이 3대를 구현하여 그림 19와 같이 배치하였다. 테스트 베드에 사용된 IPsec 공개 소프트웨어는 Free S/WAN 2.06 버전[17]을 사용하였으며 보안 프로토콜이 탑재된 게이트웨이는 리눅스 커널 2.4.30을 이용하여 구현하였다. 게이트웨이 1번을 목적지로 해서 게이트웨이 2번과 3번은 본사와 지사를 연결하는 가상 사설망을 시뮬레이션 하도록 하였다. 목적지인 1번 게이트웨이에는 2개의 네트워크 인터페이스를 구현하여 1번 인터페이스와 2번 게이트웨이는 기존의 IKE 프로토콜을 이용한 IPsec SA 터널[19]를 구성하도록 하고, 2번 인터페이스와 3번 게이트웨이는 본 연구에서 구현한 IKEv2 프로토콜을 이용하여 IPsec SA 터널을 구성하도록 하였다. 각 구간에서 동작하는 IPsec 프로토콜은 서로 다른 IKE 엔진을 탑재하도록 하여 1번 게이트웨이는 서로 다른 두개의 키 교환 엔진을 탑재한 2개의 커널로 구성되어 있고, ①번과 ②번 구간의 키 교환 성능을 테스트 할 때 마다 2번 게이트웨이 혹은 3번 게이트웨이와 연동되는 커널로 시스템을 시작하여 테스트 하도록 하였다. 그리고 각 게이트웨이 구간의 통신 패킷을 스니핑 하여 패킷을 분석하고, IPsec SA 터널을 구성 타임을 체크하여 그 성능을 평가하였다.

Ⅳ. 실험결과 분석 및 성능 평가

본 연구에서 구현한 IKEv2 프로토콜은 기존의 IKE 프로토콜에 비하여 송수신자 간의 메시지 교환 횟수를 줄이고 상태 값을 재활용함으로써 보다 빠르고 효율적으로 최종 SA를 성립하고 보안 터널을 구성하도록 하였다. 그림 20은 테스트 베드 환경의 ① 번 IKE 키 교환 구간에서 키 교환 프로세스 데몬을 시작시킴과 동시에 패킷을 스니핑 한 결과이다. 관련 연구에서 설명한 바와 같이 IKE Phase 1단계의 메인 모드는 ISAKMP 메시지[20]을 교환하는 과정을 통해 SA를 협상, IKE 키 교환 인증을 수행하게 된다. 그림 20에서 데이터의 발신지로부터 목적지까지 전송되는 패킷은 IKE 키 교환을 위한 상태전이를 보여주고 있다. 기존 IKE 키 교환 프로토콜로 구현된 IPsec 터널 구간 에서는 IKE Phase 1 단계 메시지 교환[21]을 위한 발신지로부터 목적지까지 ISAKMP 메인 모드 메시지 3개의 전송이 발생됨을 볼 수 있다. 이는 발신지와 목적지간 주고받는 총 6개의 메시지 교환 중 발신지로부터 전송되는 3개의 메시지를 보여주는 것이다. IKE 프로토콜의 메시지 교환방식에는 Phase 1에 8개 Phase 2에 1개로 총 9가지가 존재한다. 이는 다양한 방식을 지원하고자 제안되었지만, 결과적으로 IKE의 복잡성을 가중시키는 단점을 초래하였다. 1단개에서 교환하는 6개의 메시지는 첫 번째와 두 번째의 메시지 교환에서 SA 협상이 수행되고, 세 번째와 네 번째 메시지에서 DH(Diffie-Hellman) 공개키 값(KE)이 교환된다. 다섯 번째와 여섯 번째 메시지 교환에서 상호 인증을 위한 과정이 수행되는데, Identity(ID)와 Signature (SIG-I/R), 그리고 선택적으로 인증서가 교환된다. 다섯 번째 여섯 번째 메시지는 이전 단계까지의 메시지를 통해 교환된 SA와 키로 암호화 된다. IKE Phase 1단계를 통해 IKE SA(Security Association)이 확립되면, 키 교환 2단계를 통해 IPsec을 위한 SA를 협상하고, 키 값과 ID를 교환 하고 암호화된 터널을 형성한다. 이 터널을 통해 암호화된 데이터가 전송되게 된다. 그림 20에서 ISAKMP 퀵 모드[22]는 암호화된 터널을 만들기 위해 발신지와 목적지 간에 전달되는 3개의 메시지 중 목적지로 전송되는 2개의 메시지를 보여주고 있다.

기존의 IKE 프로토콜 키 교환 상태 전이와의 비교를 위해 ②번 IKEv2 프로토콜 구간에서의 상태전이를 분석한 결과는 그림 21과 같다. 그림 21은 테스트 베드 환경의 IKEv2 키 교환 구간에서 키 교환

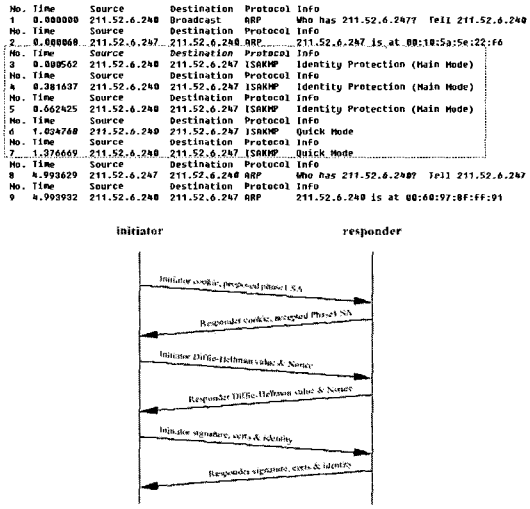

그림 20. 기존 IKE 프로토콜의 키 교환

프로세스 데몬을 시작시킴과 동시에 패킷을 스니핑 한 결과이다. 실험 결과 기존의 IKE 프로토콜에서는 1단계에서 키 교환 시 필요한 6개 메인 모드 메시지 교환을 4개의 메인 모드 메시지 교환만으로 SA를 성립할 수 있도록 메시지 교환 횟수가 줄어들었음을 알 수 있으며, 터널 형성을 위한 2단계의 퀵 모드도 2번의 메시지 교환으로 터널을 성립 할 수 있도록 개선되었음을 보이고 있다. 첫 번째 메시지 쌍에서는 IKE SA 협상 및 설정에 필요한 DH(Diffie-Hellman) 키 교환을 수행하고 Nonce 값과 필요한 파라미터들을 교환한다. 두 번째 메시지 교환에서는 상호 인증을 위한 식별정보와 인증 정보를 교환하고 Child SA 설정에 필요한 파라미터들을 협상한다. 이때 Child-SA를 협상함으로써 IPsec SA의 생성 지연을 최소화 할 수 있다. 그림 21의 결과에서 볼 수 있듯이 IKEv2 프로토콜 엔진에서는 송신지와 목적지 간에 1단계인 IKE SA를 성립 하고 2단계인 IPsec SA 터널을 구성하는데, 목적지로 보내지는 2개의 ISAKMP 메인모드 메시지와 1개의 ISAKMP 퀵 모드 메시지 전송으로 터널이 성립 되는 것을 볼 수 있다. 기존의 IKE 프로토콜과 비교 했을 때 이렇게 메시지 전송 을 통한 상태 전이의 횟수를 줄임으로써 IKEv2 키 교환 엔진에서는 기존의 키 교환 메커니즘 보다 빠르게 IKE SA와 IPsec SA 터널을 구성 하고 있음을 실험 결과 알 수 있다.

그림 22는 프로토콜 분석기를 이용하여 IKEv2 키 교환 메시지를 보다 자세하게 스니핑 하여 분석한 결과이다. 그림 22의 처음 부분은 ISAKMP 프로토콜의 헤더로서 첫 16 옥텟은 각각 8 옥텟씩의 Initiator Cookie와 Responder Cookie를 구성한다. 그 다음의 코드는 next payload, version, Exchange Type, flag등이 오고, Message ID, Message Length가 코드화 되어 있다. ISAKMP 헤더 다음에는 프로포절 페이로드와 트랜스폼 페이로드가 따라 나온다.

그림 22의 프로포절 페이로드의 프로토콜 ID는 ISAKMP이며, 트랜스폼 ID에서는 IKE 키 가 사용되고 트랜스 폼 암호화 알고리즘은 3DES Block Cypher 암호화 알고리즘[23], Hash 알고리즘에는 SHA[24], 인증방법에는 RSA-SIG[20]키가 사용됨을 볼 수 있다. 이와 같이 암호화 알고리즘을 이용해 전송되는 패킷을 분석한 결과는 그림 23과 같다. 발신지에서 목적지까지 패킷을 보낼 때, 암호화 된 패킷과 비 암호화 된 패킷은 패킷에 포함된 데이터의 내용이 패킷 스니핑을 통해서 노출되는지 안 되는지를 통해 확인 해 볼 수 있다. 암호화 된 패킷은 ESP 프로토콜에 의해 암호화 되어 패킷의 내용이 무엇인지를 패킷 스니핑을 통해서도 알 수 없으며, 단지 암호화 프로토콜이 ESP란 것과 16진수의 Security Index Policy 값만을 보여준다. 하지만 암호화를 하지 않았을 경우 그림 23의 아래 박스의 내용과 같이 발신지로부터 목적지로 보내지는 패킷이 ICMP라는 것과 ping command의 request echo를 보내고 있는 것을 확인 할 수 있다.

그림 24와 그림 25는 본 연구에서 구현한 IKEv2 프로토콜엔진과 기존의 IKE 프로토콜엔진의 최초 1단계 키 교환 메시지를 보내는 시간에서부터 SA 성립을 거쳐서 2단계 터널 성립이 완성되는데 소요되는

```
No. Time      Source       Destination  Protocol Info
1   0.000000  211.52.6.241 Broadcast    ARP      Who has 211.52.6.248?  Tell 211.52.6.241
No. Time      Source       Destination  Protocol Info
2   0.000064  211.52.6.248 211.52.6.241 ARP      211.52.6.248 is at 00:12:5b:3e:23:d6
No. Time      Source       Destination  Protocol Info
3   0.000512  211.52.6.241 211.52.6.248 ISAKMP   Identity Protection (Main Mode)
No. Time      Source       Destination  Protocol Info
4   0.351231  211.52.6.241 211.52.6.248 ISAKMP   Identity Protection (Main Mode)
No. Time      Source       Destination  Protocol Info
5   0.692723  211.52.6.241 211.52.6.248 ISAKMP   Quick Mode
No. Time      Source       Destination  Protocol Info
6   0.998461  211.52.6.248 211.52.6.240 ARP      Who has 211.52.6.241?  Tell 211.52.6.248
No. Time      Source       Destination  Protocol Info
7   1.246321  211.52.6.241 211.52.6.248 ARP      211.52.6.241 is at 00:51:93:7F:df:93
```

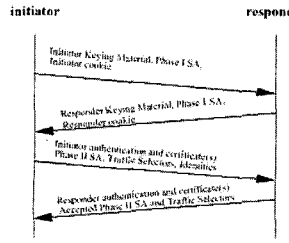

그림 21. IKEv2 프로토콜의 키 교환

```
Internet Security Association and Key Management Protocol
    Initiator cookie: 0x2BABB4659EE6404F
    Responder cookie: 0x0000000000000000
    Next payload: Security Association (1)
    Version: 1.0
    Exchange type: Identity Protection (Main Mode) (2)
    Flags
    Message ID: 0x00000000
    Length: 176
    Security Association payload
        Next payload: NONE (0)
        Length: 140
        Domain of interpretation: IPSEC (1)
        Situation: IDENTITY (1)
        Proposal payload # 0
            Next payload: NONE (0)
            Length: 136
            Proposal number: 0
            Protocol ID: ISAKMP (1)
            SPI size: 0
            Number of transforms: 4
            Transform payload # 0
                Next payload: Transform (3)
                Length: 32
                Transform number: 0
                Transform ID: KEY_IKE
                Life-Type (11): Seconds (1)
                Life-Duration (12): Duration-Value (3600)
                Encryption-Algorithm (1): 3DES-CBC (5)
                Hash-Algorithm (2): SHA (2)
                Authentication-Method (3): RSA-SIG (3)
                Group-Description (4): Alternate 1024-bit MODP group (2)
```

그림 22. ISAKMP 패킷 분석

No.	Time	Source	Destination	Protocol	Info
70	93.647812	211.52.6.248	211.52.6.247	ESP	ESP (SPI=0xd4949f02)
No.	Time	Source	Destination	Protocol	Info
71	94.647826	211.52.6.248	211.52.6.247	ESP	ESP (SPI=0xd4949f02)
No.	Time	Source	Destination	Protocol	Info
72	95.647856	211.52.6.248	211.52.6.247	ESP	ESP (SPI=0xd4949f02)
No.	Time	Source	Destination	Protocol	Info
73	96.647828	211.52.6.248	211.52.6.247	ESP	ESP (SPI=0xd4949f02)
No.	Time	Source	Destination	Protocol	Info
74	123.736851	211.52.6.248	211.52.6.247	ICMP	Echo (ping) request
No.	Time	Source	Destination	Protocol	Info
75	124.735360	211.52.6.248	211.52.6.247	ICMP	Echo (ping) request
No.	Time	Source	Destination	Protocol	Info
76	125.735332	211.52.6.248	211.52.6.247	ICMP	Echo (ping) request
No.	Time	Source	Destination	Protocol	Info
77	126.735414	211.52.6.248	211.52.6.247	ICMP	Echo (ping) request
No.	Time	Source	Destination	Protocol	Info
78	127.735325	211.52.6.248	211.52.6.247	ICMP	Echo (ping) request
No.	Time	Source	Destination	Protocol	Info
79	128.735338	211.52.6.248	211.52.6.247	ICMP	Echo (ping) request

그림 23. 암호화, 비 암호화 패킷 비교

시간을 체크 하여 속도차이가 어느 정도인지를 실험으로 검증한 결과이다. 총 시도 횟수 20번씩을 주어서 초기 단계와 2번째 단계를 서로 비교해 보았다. 테스트 베드의 실험환경은 100M 이더넷으로 구성된 LAN 환경에 각 게이트웨이간의 거리는 3m 이다. 각각의 IKE 프로토콜 버전의 1단계에 대해서 비교 실험을 한 후 나타난 결과는 그림 24와 같다.

1단계 메시지 교환에서는 많은 메시지들이 전송되며, 암호 알고리즘에서도 보안상의 이유로 복잡한 알고리즘을 쓰기 때문에 2단계 메시지 교환에 비해서 많은 시간이 걸리는 것을 볼 수 있다. IKEv2 Phase 1 단계의 메시지 교환 속도를 보면 기존의 IKE Phase 1 단계보다 절반 정도의 시간 차이가 나도록 빠르게 교환됨을 실험 결과 알 수 있다. 그림 25는 각각의 IKE 프로토콜 Phase 2단계에 대해서 비교 실험한 결과를 보여 주고 있다. 1 단계에 비해서 전송되는 메시지의 양도 적으며 암호화 알고리즘도 초기단계에 비해서 훨씬 간단한 것을 쓰기 때문에 속도는 전체적으로 빠르게 나온다. 하지만 키 교환 2단계에서도 기존의 IKE와 본 연구에서 구현한 IKEv2의 속도차이는 확실히 구분 할 수 있을 정도로 나타났으며, 성능 면에서 확실히 향상되었음을 보여주고 있다.

V. 결 론

본 논문에서 구현한 IKEv2 프로토콜은 정상 상태에서 4개의 메시지 교환에 의해 통신 쌍방이 인증된 보안채널을 수립하도록 하여, 기존 IKE 프로토콜의 6개의 메시지 교환에 의한 보안 채널을 구성 하는 상태 전이를 줄이고 간소화 하여 Security Association을 성립하는데 걸리는 시간을 대폭 단축하여 성능을 향상 시켰다. 이것은 IKEv2 RFC 문서에서 권고하는바 데로 기존 IKE Phase 1,2단계의 분리 개념을 계승하여, 1단계에서 수립된 IKE SA를 후속 IPsec SA들의 수립과 관리 시 제어 채널로 활용하고 1단계의 상태 정보를 2단계에서도 활용 할 수 있도록 설계함으로써 가능 하였다. 본 연구에서 구현한 IKEv2 프로토콜은 자주 사용되지 않는 옵션들을 없애서 프로토콜을 단순화 하여 기존의 프로토콜보다 훨씬 간단하게 하였고, 본 연구에서 테스트 베드로 사용한 공개용 IPsec VPN에 연동시킬 수 있었듯이, 멀티 벤더로 구축될 여러 종류의 IPsec 관련 어플리케이션과도 상호 연동성을 크게 향상 시킬 수 있음을 보였다.

본 연구에서 구현한 보안 프로토콜을 커널에 삽입하여 컴파일 한 IKE 시스템 엔진의 크기는 928K byte에 불과 하였다. 이는 이동 단말기와 같은 CPU와 메모리에 제약을 받는 소용량 컴퓨팅 능력을 갖는 소형 단말기에서도 보안 프로토콜 설치를 가능하게 하여 모바일 통신에서도 보안 채널을 이용한 접속 서비스를 구현하는 것이 충분히 가능하다는 것을 확인하였다. 향후 연구에서는 테스트 베드 환경을 무선 단말기 환경으로 구성하여, 무선 통신 환경에서의 보안 프로토콜에 연동되는 키 교환 프로토콜의 구현 및 안정성을 분석하고, 제반 성능을 평가하는 연구를 수행 할 예정이다.

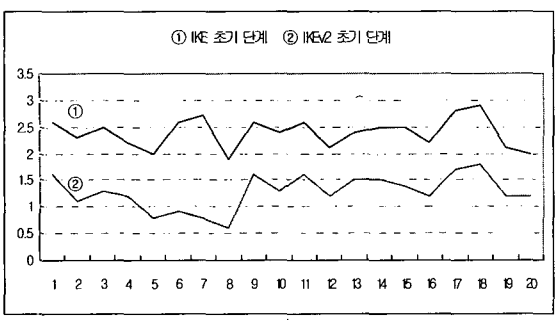

그림 24. 키 교환 초기 단계 Elapse Time

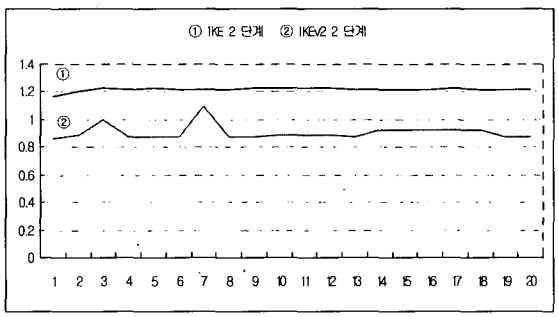

그림 25. 키 교환 2단계 Elapse Time

참 고 문 헌

[1] 김윤희, 이계상, "IPsec 테스트 베드의 구성 및 암호화 식별 도구 개발," 한국통신학회논문지 제28권 6C호 2003. 6

[2] S. Kent, et, al. "IP Authentication Header," RFC2402, IETF 1998.11

[3] S. Kent, et, al. "Encapsulating Security Payload," RFC2406, IETF 1998. 11

[4] S. Kent, et, al. "Security architecture for the Internet Protocol," RFC2401, IETF 1998. 11

[5] D. Harkins, et, al. "The Internet Key Exchange," RFC2406, IETF, 199811

[6] D. Maughan, et, al. "Internet Security Association and Key Management Protocol," RFC2408, IETF 1998.11

[7] C. Kaufman, ed. Microsoft, "Internet Key Exchange (IKEv2) Protocol," RFC4306, IETF 2005. 12

[8] 박수진, 서승현, 이상욱, "M-Commerce 사용자를 위한 효율적인 패스워드 기반 인증 및 키 교환 프로토콜," 정보과학회논문지 시스템 및 이론 제 32권 제 3호 2005. 4.

[9] 변해선, 이미정, "성형 VPN 구조에서의 주문형 터널 생성 메커니즘," 정보과학회논문지 정보통신 제 32권 제 4호 2005. 8

[10] 이재형, 김태형, 한규필, 김영학, "무선 랜에서 빠른 재 인증을 이용한 간소화된 키 관리 기법," 정보과학회논문지 정보통신 제32권 제 3호 2005. 6

[11] 이성운, 김현성, 유기영, "패스워드 기반의 효율적인 키 교환 프로토콜," 정보과학회논문지 정보통신 제 31권 제4호 2004. 8

[12] 류은경, 윤은준, 유기영, "공유 패스워드를 이용한 클라이언트 서버 인증키 교환 프로토콜," 정보과학회논문지 정보통신 제 31권 제 3호 2004. 6

[13] 이형규, 나재훈, 손승원, "IPsec 시스템에서 IKE 프로토콜 엔진의 연동에 관한 연구," 정보보호학회논문지 제 12권 제 5호, 2002. 10

[14] 이광수, 신은경, 이홍섭, "IPsec 제품의 적합성 및 상호 운용성 시험," 정보보호학회지 제 11권 제 2호 2001. 4

[15] 주한규, "IPsec의 키 교환 방식에 대한 안정성 분석," 정보보호학회논문지 제 10권 제 4호 2000. 12

[16] 최인석, 정수환, 김영한, 박용석, "IPv6 전환 기술 중 NAT-PT에서의 IPsec 적용 방안," 한국통신학회논문지 제 30권 제 11호 2005. 11

[17] www.freeswan.org

[18] T.Kivinen, H. Tschofenig, "Design of the MOBIKE protocol," Internet Draft draft-ietf-mobike-design-08 IETF 2006. 3

[19] IEEE. Standard Specifications for Public Key Cryptography, IEEE1363, 2002.

[20] V. Boyko, P. MacKenzie and S. Patel. "Provably Secure Password Authenticated Key Exchange Using Diffie-Hellman," Advances in Cryptology-EUROCRYPT'2000, 2000.

[21] P. MacKenzie, S. Patel, and R. Swaminathan, "Password authenticated key exchange based on RSA." In ASIACRYPT2000, 2000.

[22] M. Bellare and P. Rogaway, "The Authentication protocol for password-based authenticated key exchange," Presented to IEEE P1363a, March 2000.

[23] Man Li, "Policy Based IPsec Management," IEEE Network Vol. 17 No.6 pp 35 ~ 43, 2003.

[24] Y. Yang, C. U. Martel, S.F Wu, "On Building the Minimum Number of Tunnels: An Ordered-Split approach to management IPsec VPN Policies." NOMS-IEEE/IFIP Network Operations and Management Symposium, Vol. 9, No. 1, pp277 ~ 290 2004.

〈著者紹介〉

김 성 찬 (Sung-Chan Kim) 정회원
2002년 2월: 숭실대학교 정보과학 대학원 석사
2005년 2월: 숭실대학교 대학원 컴퓨터학과 박사 수료
2000년 10월~현재: (주)유코레일 전산실 차장
〈관심분야〉 정보보호, 유, 무선 통신공학
e-mail: viper72@chol.com

천 준 호 (Jun-Ho Chun) 정회원
2003년 2월: 숭실대학교 컴퓨터 학과 학사
2005년 2월: 숭실대학교 컴퓨터 학과 석사
2005년 2월~현재: 숭실대학교 컴퓨터 학과 박사과정
〈관심분야〉 네트워크 보안, 무선 라우팅 보안, 암호학
e-mail: opendream@hanmail.net

전 문 석 (Moon-Seog Jun) 종신회원
1986년 2월: University of Maryland 전산학 석사
1989년 2월: University of Maryland 전산학 박사
1989년: Morgan Sate University 전산학과 조교수
1989년~ 1991년: New Mexico Sate University 부설 Physical Science Lab 책임연구원
1991년~ 현재: 숭실대학교 정보과학대학 정교수
〈관심분야〉 네트워크 보안, 컴퓨터 알고리즘, 암호학
e-mail: mjun@computing.ssu.ac.kr