멀티미디어 정보 보안을 위한 SCPR의 설계 및 구현

홍 종 준 *, 이 재 용 **
* 청강문화산업대학대학 컴퓨터소프트웨어과
** 한서대학교 컴퓨터통신공학과

요 약

멀티미디어 정보 전송을 위해 RTP(Real-Time Transport Protocol)을 사용하며, 멀티미디어 정보 보안을 위해서는 RPT의 페이로드를 암호화해야 한다. RPT페이로드의 암호화를 위해서 암호화/복호화로 인한 지연이 멀티미디어 정보의 실시간 제약에 미치는 영향을 최소로 하면서 네트워크의 트래픽과 부하에 적응하며 암호화 알고리즘을 변경하기 위한 방법이 필요하다. 또한 멀티미디어 서비스 진행 중 서비스 이용을 중지한 사용자는 RPT 페이로드의 암호화 키를 알고 있기 때문에 페이로드를 보호하기 위해서는 암호화 키를 변경하기 위한 방법이 필요하다. 따라서 본 논문에서는 RPT 페이로드의 암호화를 위해, 암호화 알고리즘과 암호화 키를 변경하기 위한 SCPR(Security Control Protocol for RTP)를 설계하고 구현하였다.

Design and Implementation of SCPR for Multimedia Information Security

Hong, Jong-Joon*, Lee, Jae-Yong**

ABSTRACT

Real-Time Protocol (RTP) is used for multimedia information transmission and RTP payload must be encrypted for providing multimedia information security. Encryption/decryption delay is minimized, because there are constraints in transporting a multimedia data through the Internet. Therefore, encryption algorithm is changed with considering network traffic and load. During many users participate in the same multimedia service, an user who already left the service can receive and decrypt the RTP payload because of knowing the encryption key. In this paper, Security Control Protocol for RTP is designed and implemented for changing the encryption algorithm and the key.

1. 서 론

현재의 인터넷은 멀티미디어 정보의 보호에 대한 어려움으로 인해, 주문형 비디오 서비스나 비공개 화상 회의와 같은 멀티미디어 정보의 보호를 위해 RPT (Real-Time Protocol)의 페이로드를 암호화해야 한다. RPT 페이로드의 암호화는 암호화/복호화에 따른 시간 지연으로 멀티미디어 정보의 실시간 특성에 영향을 미친다. 따라서 멀티미디어 정보의 실시간 특성에 미치는 영향을 최소로 하면서 암호화를 진행하기 위해서는 네트워크의 트래픽과 부하에 적응하면서 암호화 알고리즘을 변경하기 위한 방법이 필요하다.

또한 다수가 참여하는 멀티미디어 정보의 서비스 진행 중에 서비스 이용을 중지한 사용자는 사용한 암호화 키를 알고 있기 때문에 사용된 RPT 페이로드를 수신하여 복호화할 수 있다. 그러므로 멀티미디어 서비스 진행 중에 서비스 이용을 중지한 사용자로부터 멀티미디어 서비스를 보호하기 위해 키를 변경하는 방법이 필요하다[2,7,9].

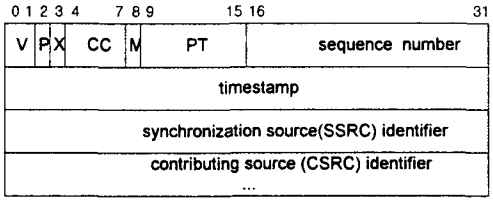
본 논문에서는 멀티미디어 정보 보호를 위한 RPT 페이로드의 암호화를 위해 암호화 알고리즘과 암호화 키를 변경하기 위한 프로토콜인 SCPR(Security Control Protocol for RTP)을 제안한다.

본 논문의 구성은 2장에서 관련연구로서 RPT의 암호화에 대한 고려사항에 대해 살펴보고, 3장에서는 제안한 SCPR의 데이터 구조와 동작 절차를 살펴본다. 4장에서는 SCPR 서버와 클라이언트의 구현 환경 및 실험 결과를 나타내며, 마지막으로 5장에서는 결론을 제시한다.

2. 관련 연구

2.1 RPT 헤더 필드

RTP는 패킷의 송신자를 구별하기 위해 RTP 패킷의 헤더에 있는 SSRC (Synchronization Source) 식별자(identifier)를 사용한다. 또한 멀티캐스트 그룹의 수신자로부터 QoS 피드백을 제공하여 대역폭을 측정할 수 있도록 하여, 트랜슬레이터(translator)와 믹서(mixer)는 회의의 모든 사용자를 수용하기 위해 요구되는 대역폭을 낮추기 위해 피드백 측정을 사용할 수 있다. 트랜슬레이터는 고품질의 스트림을 이용할 수 있는 대역폭에 적합하도록 저품질의 스트림으로 변환한다. 다수의 화상 스트림을 인터넷을 통해 전송하기 위해 각 스트림의 대역폭을 줄이는 트랜슬레이터에 보낸다. 트랜슬레이터는 변환 후에 스트림의 동기화를 보존한다. 믹서는 품질의 손실 없이 대역폭을 줄이기 위해 다수의 스트림을 하나의 스트림으로 합칠 수 있다. 믹서는 새로운 SSRC 식별자가 된다[9]. 그림 1과 같이 헤더 상위 12바이트는 모든 패킷에 나타나지만 CSRC(contributing source) 리스트는 믹서에 의해 삽입될 때만 나타난다. 각 필드에 대한 내용은 다음과 같다[1,4].



(그림 1) 실시간 전송 프로토콜 헤더

2.2 실시간 전송 프로토콜 데이터의 암호화

인터넷에서 멀티미디어 서비스를 하는 경우 전송 프로토콜로 UDP 최상위에 RTP를 적용하여 사용하기 때문에 인증 받지 않은 사용자가 전송 경로 중간에서 아무런 제약 없이 데이터를 받아 볼 수 있다. 그러나 RTP에는 전송되는 데이터를 보호하기 위한 방법이 명시되어 있지 않으므로 전송되는 데이터를 보호하기 위한 암호화가 필요하다. 멀티미디어 정보의 실시간 특성에 영

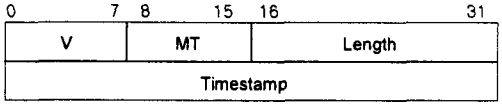
향을 최소로 하기 위해서는 대칭키 알고리즘을 사용한다[3,5,6]. 그러나 동일한 암호화 알고리즘과 동일한 암호화 키를 사용하여 페이로드를 암호화하는 경우 페이로드의 암호화/복호화로 인한 일정한 지연이 추가된다. 이러한 지연은 네트워크 트래픽과 부하에 따라 멀티미디어 정보의 실시간 제약에 많은 영향을 준다. 또한 멀티미디어 서비스 이용을 중지한 사용자가 페이로드의 암호화 키를 알고 있기 때문에 서비스 이용을 중지한 사용자로부터 멀티미디어 서비스를 보호하지 못하는 문제점이 있다. 따라서 문제를 해결하기 위해서는 암호화 알고리즘과 키를 변경하기 위한 방법이 필요하다.

3. SCPR(Security Control Protocol for RPT) 제안

제안한 SCPR은 RTP 세션에 참여하려는 사용자들과 TCP 연결을 설정하고, SCPR의 채널을 보호하기 위해 SCPR을 사용하는 응용이 암호화 알고리즘을 선택하여 전체 데이터를 암호화하거나 TLS(Transport Layer Security)[4]을 사용하여 SCPR의 채널을 보호한다. SCPR의 채널 보호는 세션 보호와 밀접한 관계가 있고 SCPR의 데이터는 실시간 제약성을 갖지 않으므로 SCPR 채널은 암호화 알고리즘에 의해 보호된다.

3.1 제안한 SCPR의 헤더

SCPR의 헤더는 그림 2와 같고 각 필드에 대한 설명은 다음과 같다.



(그림 2) SCPR 헤더

버전(V)은 현재 제안한 SCPR의 버전을 나타내고, 메시지 타입(MT)은 전송되는 메시지의 타입을 나타내며 각 타입은 3.2절에서 정의한다. 또한 길이(Length)는 헤더를 제외한 실제 데이터의 바이트 수이다. 타임스탬프(Timestamp)는 패킷 구성 시간을 나타내며 패킷의 라운드 트립 시간 계산과 재전송 공격[8]을 막기 위해 사용된다.

3.2 제안한 SCPR의 데이터 타입

대부분의 데이터 구조의 크기는 고정되어 있고, 고정되지 않은 Init Info, Key Change 데이터 구조는 Key Length 필드를 참조하여 실제 크기를 알 수 있기 때문에 패딩 여부와 패딩 비트의 크기는 헤더의 Length 필드와 데이터 구조의 종류에 의해 알 수 있다. 각 타입은 표 1과 같고 각 타입에 대한 설명은 다음과 같다.

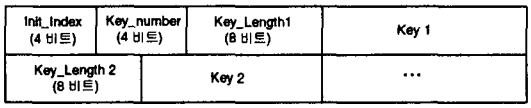
<표 1> SCPR의 데이터 타입

번호	메시지 타입	메시지 진행 방향
0	Authentication	
1	SSRC Info	서버->클라이언트
2	Init Info	서버->클라이언트
3	Algorithm Change Request	클라이언트->서버
4	Algorithm Change	서버->클라이언트
5	Key Change	서버->클라이언트
6	Packet Ack	클라이언트->서버
7	Session Leave Request	클라이언트->서버
8	Session Leave	서버->클라이언트
9	Session End	서버->클라이언트

- **Authentication** : 인증 과정에서 SCPR 채널 보호를 위해 사용되는 암호화 알고리즘의 암호화 키를 받아온다.

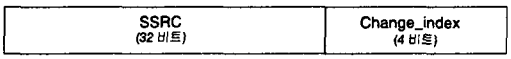
- **SSRC Info** : SSRC Info는 세션에서 사용할 클라이언트의 SSRC 식별자를 전달한다.

- Init Info : Init Info는 페이로드의 보호를 위해 사용하는 암호화 알고리즘에 의해 사용될 암호화 키와 시작 알고리즘의 인덱스 번호를 사용자들에게 전달하는 메시지 타입으로 데이터 구조는 다음과 같다.



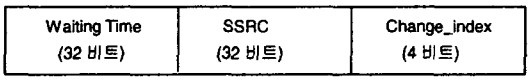
(그림 3) Init Info의 데이터 타입

- Algorithm Change Request : 클라이언트가 자신이 사용하는 암호화 알고리즘을 변경하고자 할 때 서버에 전송하는 데이터 타입으로 데이터 구조는 다음과 같다.



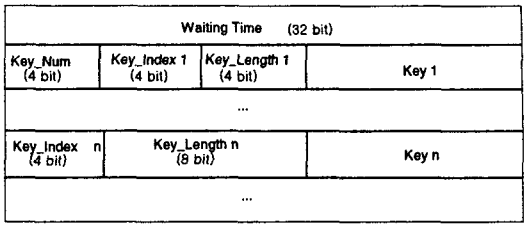
(그림 4) Algorithm Change Request 데이터 구조

- Algorithm Change : Algorithm Change는 모든 클라이언트들에게 특정 클라이언트나 모든 클라이언트에서 사용하는 암호화 알고리즘의 변화를 알려주기 위해 사용되며 데이터 구조는 그림 5와 같다.



(그림 5) Algotithm Change의 데이터 구조

- Key Change : Key Change는 암호화 알고리즘에서 사용하는 암호화 키를 변경하기 위해 사용되며 데이터 구조는 그림 6과 같다. Key-Num은 변경하고자 하는 암호화 키의 개수를 나타내며, Key-Index는 암호화 알고리즘의 인덱스 값, Key-Length는 Key 필드의 길이, Key는 변경할 키를 나타낸다.



(그림 6) Key Change의 데이터 구조

- Packet Ack : Packet Ack는 서버와 각 클라이언트 사이의 라운드 트립 시간을 측정하기 위해 특정 SCPR 패킷을 받은 모든 클라이언트에 의해 서버로 보내지며 데이터 구조는 그림 7과 같다. MT는 이 패킷이 어떤 패킷에 대한 응답인지를 나타내기 위해 사용되는 필드로 Init Info, Algorithm Change, Key Change의 값을 가질 수 있다.

(그림 7) Packet Ack의 데이터 구조

- Session Leave Request : 클라이언트가 세션을 떠나기 전에 서버에 전달하는 메시지로 사용시간으로 요금을 부과하기 위한 목적으로 사용될 수 있다.

- Session Leave : 서버가 특정 클라이언트의 세션 나감을 다른 클라이언트들에게 전달하는 메시지를 나타낸다.

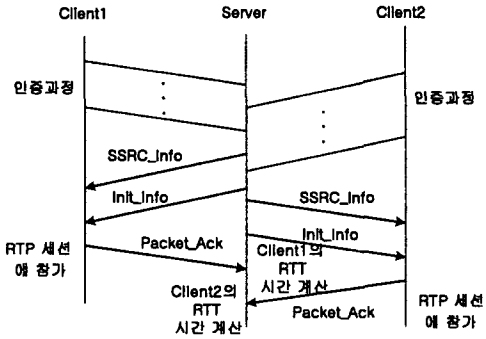
- Session End : 세션이 종료될 때 서버가 현재 세션에 참여하고 있는 모든 클라이언트에게 보내는 메시지를 나타낸다.

3.3 제안한 SCPR의 동작 절차

(1) 클라이언트가 세션에 참여하기 전의 절차

세션에 참여하기 전에 클라이언트는 인증을 받은 후, SSRC Info와 Init Info 패킷을 서버로부터 받아 SSRC 식별자, 사용할 암호화 알고리즘

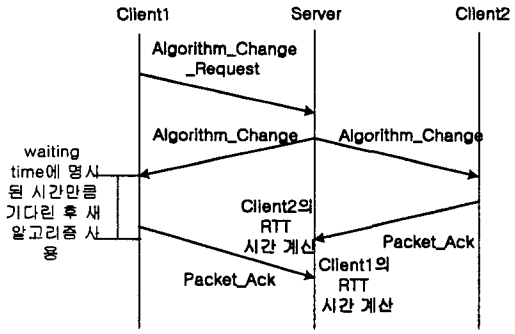
의 인덱스와 각 암호화 알고리즘에서 사용할 암호화 키 값을 얻는다. 클라이언트는 Init Info 패킷을 받은 후 서버에 Packet Ack 패킷을 보내 라운드 트립 시간을 계산하도록 한후 클라이언트는 세션에 참여한다.



(그림 8) 클라이언트가 RTP 세션에 참여하기 전의 절차

(2) 클라이언트의 암호화 알고리즘 변경 절차

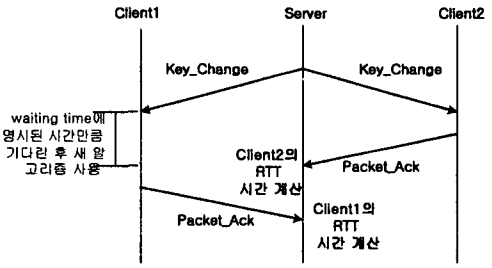
암호화 알고리즘을 변경하려는 클라이언트는 서버에 Algorithm Change Request 패킷을 보내고 서버는 모든 클라이언트에게 Algorithm Change 패킷을 보낸다. 클라이언트는 서버에 Packet Ack를 보내고 서버는 라운드 트립시간을 계산한다. 클라이언트가 Algorithm Change 패킷을 받고 Waiting Time 만큼 기다린 후 새 암호화 알고리즘을 사용한다.



(그림 9) 클라이언트의 암호화 알고리즘 변경 절차

(3) 암호화 키 변경 절차

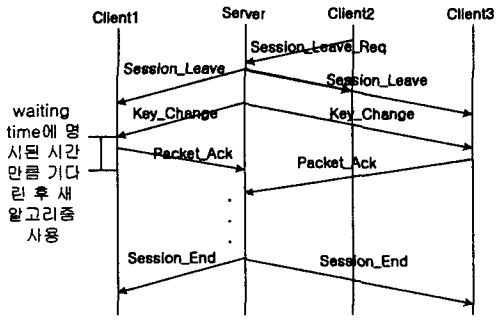
암호화 키를 변경하는 경우에는 서버가 모든 클라이언트에게 Key Change 패킷을 전송하고 클라이언트는 서버에 Packet Ack를 보낸다. 서버는 라운드 트립 시간을 계산한다. 클라이언트는 Key Change 패킷을 받고 Waiting Time 만큼 기다린 후 새 암호화 키를 사용한다.



(그림 10) 암호화 키 변경 절차

(4) 세션 종료와 클라이언트가 세션을 떠나는 절차

세션을 떠나려는 클라이언트는 서버에 Session Leave Request 패킷을 보낸다. 서버는 클라이언트들에게 Session Leave 패킷을 보내 특정 클라이언트가 세션을 떠났다는 것을 알려준 후 클라이언트들에게 Key Change 패킷을 보내 암호화 키를 변경하도록 한다. 새 라운드 트립 시간을 계산하기 위해 Key Change 패킷을 받은 클라이언트들은 서버에 Packet Ack 패킷을 보낸다. 또한 세션이 끝난 경우 서버는 세션에 참여한 모든 클라이언트들에게 Session End 패킷을 보내 RTP 세션이 끝났다는 것을 알려준다. 메시지 흐름은 그림 11과 같다.

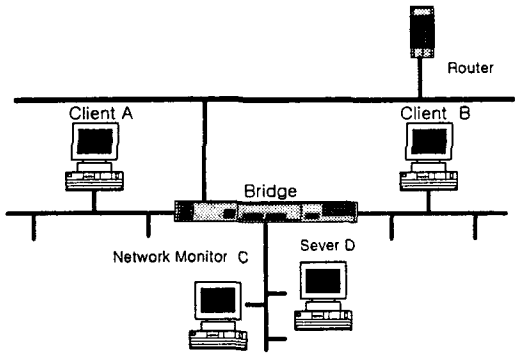


(그림 11) 세션 종료와 클라이언트가 세션을 떠나는 경우

4. SCPR 구현 및 실험 결과

4.1 구현 환경

데이터의 암호화를 확인하기 위하여 그림 12와 같은 실험 환경을 구축하였다. 3개의 다른 LAN에 SCPR 서버, 클라이언트를 각각 설치하고 모니터 호스트를 이용하여 전송 결과를 확인하였다.



(그림 12) 실험 환경

제안된 모델에서 트래픽이 서버의 상태와 클라이언트의 요청에 의해 어떻게 변화하는지를 보기 위해 모니터 호스트를 두었다. 모니터 호스트에서는 Van Jacobsen's tcpdump 프로그램이 실행되었다.

4.2 구현 및 실험 결과

구현한 SCPR 서버와 클라이언트의 동작을 확인하기 위하여 TCP 패킷을 dump한 결과는 다음과 같다.

다음 그림 13는 SSRC 클라이언트 B가 SSRC 서버 C에 연결하는 절차를 보인다.

단계 1 : B가 C에 TCP연결설정을 한다.

14:05:47.911719 B.2040 > D.500: S 153355416:153355416(0) win 8192 <mss 1460> (DF)

14:05:47.911912 D.500 > B.2040: S 53226:53226(0) ack 153355417 win 8760 <mss 1460> (DF)

14:05:47.912328 B.2040 > D.500: . ack 1 win 8760 (DF)

단계 2 : D에 연결된 B에게 SSRC 식별자를 전송한다.

14:05:47.914270 D.500 > B.2040: P 1:13(12) ack 1 win 8760 (DF)

14:05:48.039896 B.2040 > D.500: . ack 13 win 8748 (DF)

단계 3 : D에 연결된 B에게 7암호화 키와 시작 알고리즘에 대한 정보를 전송하고 B는 응답 메시지를 D에 전송한다.

14:05:48.040088 D.500 > B.2040: P 13:77(64) ack 1 win 8760 (DF)

14:05:48.042199 B.2040 > D.500: P 1:21(20) ack 77 win 8684 (DF)

14:05:48.235110 D.500 > B.2040: . ack 21 win 8740 (DF)

(그림 13) SSRC Info, Init Info의 실험결과

다음 그림 14는 B의 요청에 의해 암호화 알고리즘을 변경하는 절차를 보인다.

단계 1 : B가 D에게 알고리즘 변경 요청을 한다.
14:05:58.657415 B.2040 > D.500: P 21:37(16) ack 77 win 8684 (DF)
단계 2 : D는 모든 클라이언트들에게 알고리즘 변경을 알린다.
14:05:58.658145 D.500 > B.2040: P 77:97(20) ack 37 win 8724 (DF)
14:05:58.658200 D.500 > A.2775: P 77:97(20) ack 21 win 8716 (DF)
단계 3 : 클라이언트들이 D에게 응답 메시지를 보낸다.
14:05:58.659118 A.2775 > D.500: P 21:41(20) ack 97 win 8640 (DF)
14:05:58.659530 B.2040 > D.500: P 37:57(20) ack 97 win 8664 (DF)
14:05:58.844028 D.500 > A.2775: . ack 41 win 8696 (DF)
14:05:58.844087 D.500 > B.2040: . ack 57 win 8704 (DF)

(그림 14) Algorithm Change Request와 Algorithm Change의 실험결과

그림 15는 암호화 키를 변경하는 절차를 보인다.

단계 1 : D가 클라이언트들에게 키 변경을 알린다.
14:06:08.314173 D.500 > B.2040: P 97:165(68) ack 57 win 8704 (DF)
14:06:08.314250 D.500 > A.2775: P 97:165(68) ack 41 win 8696 (DF)
단계 2 : 클라이언트들이 D에게 응답메세지를 보낸다.
14:06:08.315191 A.2775 > D.500: P 41:61(20) ack 165 win 8572 (DF)
14:06:08.315919 B.2040 > D.500: P 57:77(20) ack 165 win 8596 (DF)
14:06:08.468622 D.500 > A.2775: . ack 61 win 8676 (DF)
14:06:08.468684 D.500 > B.2040: . ack 77 win 8684 (DF)

(그림 15) Key Change의 실험결과

그림 16는 B가 세션을 떠나는 절차를 보인다.

단계 1 : B가 D에게 Session Leave Request 메시지를 전송한다.
14:06:14.620299 B.2040 > D.500: P 77:89(12) ack 165 win 8596 (DF)
단계 2 : D는 모든 클라이언트들에게 B가 세션에서 떠난다는 걸 알려주기 위해 Session Leave 메시지를 전송하고 B와의 TCP 연결을 해지한다.
14:06:14.620929 D.500 > B.2040: P 165:177(12) ack 89 win 8672 (DF)
14:06:14.620984 D.500 > A.2775: P 165:177(12) ack 61 win 8676 (DF)
14:06:14.621253 D.500 > B.2040: F 177:177(0) ack 89 win 8672 (DF)
14:06:14.621602 B.2040 > D.500: . ack 178 win 8584 (DF)
14:06:14.804743 A.2775 > D.500: . ack 177 win 8560 (DF)
단계 3 : D는 암호화 키를 변경하기 위해 남은 클라이언트들에게 Key Change 메시지를 전송한다.
14:06:14.804920 D.500 > A.2775: P 177:245(68) ack 61 win 8676 (DF)
14:06:14.805821 A.2775 > D.500: P 61:81(20) ack 245 win 8492 (DF)
14:06:14.921479 D.500 > A.2775: . ack 81 win 8656 (DF)

(그림 16) Session Leave Request, Session Leave의 실험결과

구현 결과, 클라이언트의 암호화 알고리즘 변경 요청에 의해서 서버는 모든 클라이언트들에게 특정 클라이언트의 암호화 알고리즘의 변화를 전달됨을 확인하였다. 또한, 특정 클라이언트가 멀티미디어 서비스 진행 중에 서비스 이용을 중지하는 경우 SCPR 서버가 다른 클라이언트에서 사용하는 암호화 키를 변경함을 확인하였다.

6. 결 론

본 논문에서는 RTP을 위한 암호화 알고리즘과 암호화 키를 변경할 수 있는 SCPR을 제안하

였다. 클라이언트가 네트워크의 트래픽이나 부하로 인해 사용하고 있는 암호화 알고리즘이 적합하지 않다고 판단되면 암호화 알고리즘을 선택하여 서버에 Algorithm Change Request 메시지를 보내고 서버는 모든 다른 클라이언트들에게 Algorithm Change 메시지를 보내, 특정 클라이언트의 암호화 알고리즘의 변경을 다른 참가자들에게 전달하여 암호화 알고리즘을 변경하였다. 또한, 특정 클라이언트가 멀티미디어 서비스의 이용을 중지한 후, 서버는 Key Change 메시지를 다른 클라이언트들에게 전송하여 암호화 키를 변경하도록 하였다. 그 결과 멀티미디어 서비스 이용을 중지한 클라이언트로부터 서비스를 보호할 수 있다.

실제 네트워크에서 실험한 결과 본 논문에서 제안한 방식이 제시된 실험환경에서 암호화 알고리즘을 변경하고, 암호화 키를 변경하여 전송 데이터를 보호할 수 있음을 확인하였다.

참고문헌

[1] Sousa P. Freitas V, "A Framework for the Development of Tolerant Real-Time Application," Computer Networks & Isdn Systems, Vol. 30, No. 16-18, pp. 1531 - 1541, Sep., 1998.

[2] Chi-Sung Laith, Sung-Ming Yen, "On the Design of Conference Key Distribution Systems for the Broadcasting Networks," IEEE Infocom'93, Vol. 3, pp. 1406 - 1413, May., 1993.

[3] Aikawa M, Takaragi K, Furuya S, Sasamoto M, "Lightweight Encryption Method Suitable for Copyright Protection," IEEE Transactions on Consumer Electronics, Vol .44, No. 3, pp. 902-910, Aug., 1998.

[4] G. Mamais, M. Markaki. M. H. Sherif, G. Stassinopoulos, "Evaluation of the Casner-Jacobson Algorithm for Compressing the RTP/UDP/IP Headers," Proc. of the Third IEEE Symposium on Computers and Communications, pp. 543-548, Jun., 1998.

[5] Alfred J. Menezes, Paul C. van Oorschot, Scott A. Vanstone, Handbook of Applied Cryptography, CRC Press, 1996.

[6] Charles P. Pfleeger, Security Computing, Prentice Hall, 1997.

[7] Oppliger R, Albanese A, "Participant Registration, Validation, and Key Distribution for Large-Scale Conferencing Systems," IEEE Communication Magazine, Vol. 35 No. 6, pp. 130 - 134, Jun., 1997.

[8] H. Schulzrinne, "RTP : A Transport Protocol for Real-Time Application," RFC1889, Jan., 1996.

[9] Shoichiro Seno, Akira Watanabe, Yuuji Koui, Masayuki Yabe, and Tetsuo Ideguchi, "Intranet Packet Encryption with Minimum Overheads," Proc. of the 13th ICCC, pp. 517-521, Nov., 1997.

홍 종 준

1991년 인하대학교 전자계산공학과(공학사)
1993년 인하대학교 전자계산공학과(공학석사)
2002년 인하대학교 전자계산공학과(공학박사)
1999 ~ 현재 청강문화산업대학 컴퓨터소프트웨어과 교수

이 재 용

1985년 인하대학교 전자계산학과 (이학사)
1990년 인하대학교 전자계산학과(이학석사)
2000년 인하대학교 전자계산공학과(공학박사)
2000 ~ 현재 한서대학교 컴퓨터통신공학과 교수