論文
3-9-10

멀티미디어 정보 보호를 위한 RTP 보안 제어 프로토콜 설계 및 구현 (Design and Implementation of RTP Security Control Protocol for Protecting Multimedia Information)

홍 종 준*
(Jong-Joon Hong)

요 약

주문형 비디오 서비스나 비공개 화상 회의와 같은 보호가 필요한 서비스를 위해서는 RTP의 payload를 암호화해야 한다. 멀티미디어 데이터는 실시간 제약을 갖고 있기 때문에 암호화/복호화로 인한 지연이 실시간 제약에 미치는 영향을 최소로 하면서 암호화를 하기 위해서는 네트워크 트래픽과 부하에 적응하여 암호화 알고리즘을 변경하기 위한 방법이 필요하다. 또한 다수가 참여하는 멀티미디어 서비스 진행 중에 서비스 이용을 중지한 사용자는 RTP payload의 암호화 키를 알고 있기 때문에 이 사용자로부터 RTP payload를 보호하기 위해서는 암호화 키를 변경하기 위한 방법이 필요하다. 따라서 본 논문에서는 RTP payload의 암호화를 위해 암호화 알고리즘과 암호화 키를 변경하기 위한 SCPR(Security Control Protocol for RTP)를 설계하고 구현하였다.

ABSTRACT

RTP payload must be encrypted for providing commercial VOD service or private video conference over the Internet. 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. 서론

RTP(Real-time Transport Protocol)는 실시간 정보를 전송하는 응용 프로그램에서 요구되는 네트워크의 양단간 멀티미디어 데이터의 전송을 위한 기능을 제공하기 위한 프로토콜이다. RTP는 모든 수송 계층의 기능을 수용하기 보다는 응용 프로그램의 측면에서 하부 망의 기능을 수정 보완하는 기능을 가지고 있다. 즉, 별도의 계층으로 존재하기 보다는 응용 프로그램의 처리부분에 포괄적으로 포함된다. 따라서 하부망 프로토콜인 네트워크 계층의 프로토콜과는 독립적으로 동작하도록 되어있다. 일반적으로 응용

* 정회원 : 청강문화산업대학 컴퓨터소프트웨어과 교수

논문접수 : 2002. 8. 16.
심사완료 : 2002. 12. 10.

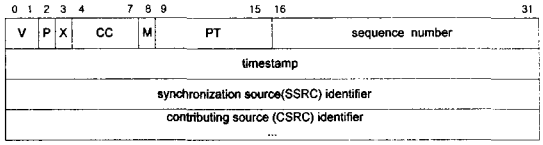
프로그램은 UDP에서 제공하는 다중화 기능 및 체크섬 기능 등을 활용하기 위해 UDP상에서 RTP를 실행시킨다. 즉, 두 개의 수송계층 프로토콜을 모두 이용하여 총괄적인 수송계층 서비스를 제공하게 된다[1]. 그러나 현재의 인터넷은 전송되는 데이터를 보호할 수 없기 때문에 주문형 비디오 서비스나 비공개 화상 회의와 같은 멀티미디어 서비스의 보호를 위해 RTP의 페이로드를 암호화해야 한다. RTP 페이로드의 암호화는 암호화/복호화하는 시간으로 인해 멀티미디어 서비스의 실시간 특성에 영향을 미친다. 따라서 멀티미디어 서비스의 실시간 특성에 미치는 영향을 최소로 하면서 암호화를 하기 위해서는 네트워크의 트래픽과 부하에 적응하면서 암호화 알고리즘을 변경하기 위한 방법이 필요하다. 또한 다수가 참여하는 멀티미디어 서비스 진행 중에 서비스 이용을 중지한 사용자는 사용한 암호화 키를 알고 있기 때문에 멀티미디어 서비스를 위해 사용된 RTP 페이로드를 수신하여 복호화할 수 있다. 그러므로 멀티미디어 서비스 진행 중에 서비스 이용을 중지한 사용자로부터 멀티미디어 서비스를 보호하기 위해 키를 변경하는 방법이 필요하다[4,9,11].

본 논문에서는 멀티미디어 정보 보호를 위해, RTP 페이로드의 암호화를 위한 암호화 알고리즘과 암호화 키를 변경하기 위한 프로토콜인 RTP 보안제어프로토콜을 제안한다.

2. 관련 연구

2.1 RTP 헤더필드

RTP는 패킷의 송신자를 구별하기 위해 RTP 패킷의 헤더에 있는 SSRC (Synchronization Source) 식별자(identifier)를 사용한다. 또한 RTP는 멀티캐스트 그룹의 수신자로부터 QoS피드백을 제공한다. 이 피드백은 송신자와 통신할 때 회의에서 소비되는 대역폭의 크기를 측정하기 위해 사용된다. 트랜슬레이터(translator)와 믹서(mixer)는 회의의 모든 사용자를 수용하기 위해 요구되는 대역폭을 낮추기 위해 피드백 측정을 사용할 수 있다. RTP의 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[14]. 그림 1과 같이 RTP의 헤더 상위 12바이트는 모든 RTP 패킷에 나타나지만 CSRC(contributing source) 리스트는 믹서에 의해 삽입될 때만 나타난다. 각 필드에 대한 내용은 다음과 같다[2,3,6].



[그림 1] RTP 헤더
[Fig. 1] RTP Header

버전(V) : 2비트. RTP의 버전. 가장 최근의 버전은 2이다.

패딩(P) : 1비트. 설정되어 있으면 패킷은 그 끝에 페이로드의 부분이 아닌 1바이트 이상의 패딩 바이트를 포함한다. 패딩의 마지막 바이트는 얼마나 많은 패딩 바이트를 무시해야 하는지에 대한 카운터이다.

확장(X) : 1비트. 설정되어 있으면 고정 헤더 뒤에 하나의 확장 헤더가 따라온다.

CSRC 카운터(CC) : 4비트. 고정 헤더 다음에 나오는 CSRC 식별자의 수이다. RTP 패킷의 페이로드가 몇몇 소스로부터 온 데이터를 가지고 있으면 이 카운터의 값은 1 이상이다.

마커(M) : 1 비트. 프로파일에 의해 정의된다. 마커는 프레임 경계와 같은 중요한 이벤트를 패킷 스트림에 표시하기 위

해 사용된다.

payload type(PT) : 7 비트. RTP 페이로드의 포맷을 나타내고 애플리케이션에 의한 해석을 결정한다.

sequence number : 16 비트. 전송되는 각 RTP 데이터 패킷에 대해서 1씩 증가한다. 패킷 손실 검출과 패킷 순서의 복구를 위해서 수신측에서 사용된다. 초기 값은 임의로 설정된다.

timestamp : 32 비트. RTP 데이터 패킷에 있는 첫 번째 바이트가 샘플링 되는 순간을 반영한다. 동기화와 지터 계산을 위해 사용된다. 초기 값은 임의로 설정된다.

SSRC : 32 비트. 같은 RTP 세션 내의 동기화 소스를 구분하기 위해 임의로 선택된 값이다.

CSRC 리스트 : 0 ~ 15 개의 아이템, 각 아이템은 32 비트. 패킷의 페이로드에 기여한 소스들을 나타낸다. 식별자의 수는 CC 필드에 주어진다.

2.2 RTP 데이터의 암호화

인터넷에서 멀티미디어 서비스를 하는 경우 전송 프로토콜로 UDP 최상위에 RTP를 적용하여 사용하기 때문에 인증받지 않은 사용자가 전송 경로 중간에서 아무 제약 없이 RTP 데이터를 받아 볼 수 있다. 그러나 RTP에는 전송되는 데이터를 보호하기 위한 방법이 명시되어 있지 않기 때문에 전송되는 RTP 데이터를 보호하기 위한 RTP 데이터의 암호화가 필요하다. 멀티미디어 데이터의 실시간 특성에 영향을 최소로 하기 위해서는 대칭키 알고리즘을 사용한다[5][7][8]. 그러나 멀티미디어 서비스의 처음부터 끝까지 동일한 암호화 알고리즘과 동일한 암호화 키를 사용하여 RTP 페이로드를 암호화하는 경우 페이로드의 암호화/복호화로 인한 일정한 지연이 추가된다. 이러한 지연은 네트워크 트래픽과 부하에 따라 멀티미디어 데이터의 실시간 제약에 많은 영향을 준다. 또한 멀티미디어 서비스 이용을 중지한 사용자가 RTP 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.

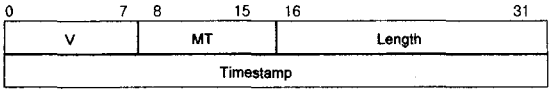
3. RTP 보안 제어 프로토콜 제안

멀티미디어 데이터는 실시간 제약을 가지기 때문에 멀티미디어 데이터의 암호화가 이런 실시간 제약에 영향을 최소로 주면서 암호화를 하기 위해서는 네트워크의 트래픽과 부하에 따라 최적의 알고리즘을 사용해서 암호화를 해야 한다. 또한 RTP 세션 도중에 세션에서 나간 사용자에게 RTP 페이로드를 보호하기 위해 암호화 키를 바꿔야 한다. 그렇지 않으면 세션에서 나간 사용자가 RTP 페이로드를 받아 볼 수 있다. 이런 문제를 해결하기 위해 본 논문에서는 암호화 알고리즘과 암호화 키를 바꾸는 메카니즘을 제공하는 RTP 보안제어프로토콜을 제안하였다.

RTP 보안 제어 프로토콜은 RTP 세션에 참여하려는 사용자들과 TCP 연결을 설정하고, RTP 보안제어프로토콜의 채널을 보호하기 위해 RTP 보안제어 프로토콜을 사용하는 응용이 암호화 알고리즘을 선택하여 전체 데이터를 암호화 하거나 TLS(Transport Layer Security)[6]을 사용하여 RTP 보안 제어 프로토콜의 채널을 보호한다. RTP 보안 제어 프로토콜의 채널 보호는 RTP 세션 보호와 밀접한 관계가 있고 RTP 보안 제어 프로토콜의 데이터는 실시간 제약성을 갖지 않으므로 RTP 보안제어프로토콜 채널은 암호화 알고리즘에 의해 보호된다.

3.1 RTP 보안 제어 프로토콜의 헤더

RTP 보안 제어 프로토콜의 헤더는 [그림 2]와 같고 각 필드에 대한 설명은 다음과 같다.



[그림 2] RTP 보안 제어 프로토콜 헤더
[Fig. 2] RTP Security Control Protocol Header

버전(V) : 1 바이트. 현재 RTP 보안 제어 프로토콜의 버전을 나타낸다.

메시지 타입(MT) : 1 바이트. 전송되는 메시지의 타입을 나타낸다. 각 타입은 3.2절에서 정의한다.

길이(Length) : 2 바이트. 헤더를 제외한 실제 데이터의 바이트 수이다.

타임스탬프(Timestamp) : 4 바이트. 패킷이 구성된 시간의 값이다. 패킷의 라운드 트립 시간 계산과 재전송 공격[10]을 막기 위해 사용된다.

3.2 RTP 보안 제어 프로토콜의 데이터 타입

RTP 보안 제어 프로토콜 헤더의 Length 필드가 데이터의 바이트 단위 길이를 나타내기 때문에 실제 데이터 필드의 길이가 바이트 단위가 아니면 데이터의 뒤에 패딩(padding) 비트가 붙어 바이트 단위의 길이로 만들어야 한다. 대부분의 데이터 구조의 길이는 고정되어 있고, 고정되어 있지 않은 Init Info, Key Change 데이터 구조도 Key Length 필드를 참조하면 실제 데이터 부분의 크기를 알 수 있기 때문에 패딩 여부와 패딩 비트의 크기는 헤더의 Length 필드와 데이터 구조의 종류에 의해 알 수 있다.

각 타입은 <표 1>과 같고 각 타입에 대한 설명은 다음과 같다.

<표 1> RTP 보안 제어 프로토콜의 데이터 타입
<Table 1> Data Type of RTP Security Control Protocol

메시지 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

(1) Authentication

사용자 인증 프로토콜에 의해 사용되고, RTP 보안 제어 프로토콜을 사용하는 응용에서 설정한다. 인증 과정에서 RTP 보안 제어 프로토콜 채널 보호를 위해 사용되는 암호화 알고리즘의 암호화 키를 받아온다.

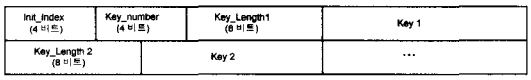
(2) SSRC Info

SSRC Info는 RTP 세션에서 사용할 클라이언트의 SSRC 식별자 를 전달하는 데이터 구조이다. SSRC 식별자를 각 클라이언트에서 임의의 값으로 선택하는 경우 RTP 세션에 참여한 클라이언트 사이에 SSRC 식별자의 충돌이 발생할 수 있다. 이와 같은 SSRC 식별자 충돌은 각 RTP 구현에서 검출하고 해결해야 한다[11]. 그러나 SCPR 서버에서 RTP 세션에 참여하려는 각 클라이언트의 SSRC 식별자를 다른 클라이언트가 사용하지 않는 임의의 값으로 선택하여 전송하면 SSRC 식별자의 충돌이 발생하지 않는다.

(3) Init Info

Init Info는 RTP 페이로드의 보호를 위해 사용하

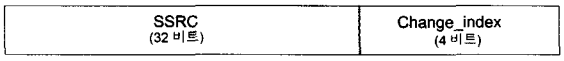
는 암호화 알고리즘에 의해 사용될 암호화 키와 시작 알고리즘의 인덱스 번호를 사용자들에게 전달하는 메시지 타입이다. Init Info의 데이터 구조는 그림 3과 같다. Init_index는 처음 RTP 페이로드를 암호화할 암호화 알고리즘에 대한 인덱스 번호이고 Key_number는 전송되는 Key_length와 Key의 개수를 나타낸다. Key의 개수와 암호화 알고리즘의 수는 같으므로 Key_number는 사용하는 암호화 알고리즘의 개수를 나타낸다. 사용되는 암호화 알고리즘의 종류는 SCPR을 사용하는 응용에서 설정한다. Key_Length n은 Key n의 길이를, Key n은 n번째 암호화 알고리즘에서 사용하는 키를 나타내고(1 ≤ n ≤ Key_number, n은 정수) Key 값의 설정은 응용에서 선택한다.



[그림 3] Init Info의 데이터 타입
[Fig. 3] Data Type of Init Info

(4) Algorithm Change Request

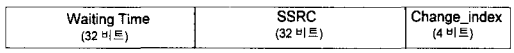
클라이언트가 자신이 사용하는 암호화 알고리즘을 변경하고자 할 때 RTP 보안 제어 프로토콜 서버에 전송하는 데이터 타입이다. Algorithm Change Request의 데이터 구조는 [그림 4]와 같다. 변경되는 암호화 알고리즘은 현재 네트워크 트래픽 상태와 부하에 따라 응용에 의해 가장 적합한 암호화 알고리즘이 선택된다. 암호화 알고리즘의 선택에 관한 세부 사항은 RTP 보안 제어 프로토콜을 사용하는 응용에 의해 구현되어야 한다. RTP 보안 제어 프로토콜 식별자는 이 데이터 타입을 보낸 송신자의 RTP 보안 제어 프로토콜 식별자이고 Change_Index는 변경하고자 하는 알고리즘의 인덱스 번호이다.



[그림 4] Algorithm Change Request 데이터 구조
[Fig. 4] Data Structure of Algorithm Change Request

(5) Algorithm Change

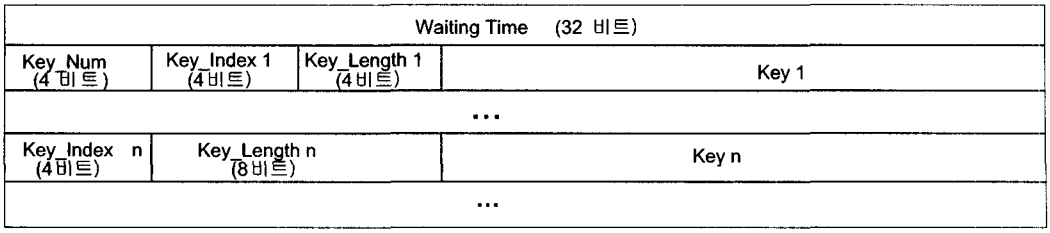
Algorithm Change는 모든 클라이언트들에게 특정 클라이언트나 모든 클라이언트에서 사용하는 암호화 알고리즘의 변화를 알려주기 위해 사용된다. Algorithm Change의 데이터 구조는 [그림 5]와 같다. Waiting Time은 새로운 암호화 알고리즘의 사용 시점을 동기화 하기 위해 사용된다. 클라이언트들은 Algorithm Change 메시지를 받은 시점부터 Waiting Time의 시간만큼 기다린 후에 Change_Index에 나타난 암호화 알고리즘을 사용한다. Waiting Time은 각 클라이언트의 라운드 트립 시간 중 가장 큰 값에서 각 클라이언트의 라운드 트립 시간을 뺀 값이다. 이 값은 각 클라이언트로 보내지는 패킷마다 다른 값을 가진다. 멀티미디어 데이터는 일부 데이터의 손실이 있어도 문제가 되지 않으므로 새 알고리즘 시작의 동기화가 정확하게 이루어지지 않아 몇 개의 RTP 패킷이 손실되더라도 문제가 되지 않는다. SSRC 필드는 암호화 알고리즘을 변경하려는 클라이언트의 SSRC 식별자이다. 이 값이 0이면 모든 클라이언트가 현재 사용하고 있는 암호화 알고리즘을 Change_Index에 명시된 알고리즘으로 바꾼다.



[그림 5] Algotithm Change의 데이터 구조
[Fig. 5] Data Structure of Algorithm Change

(6) Key Change

Key Change는 암호화 알고리즘에서 사용하는 암호화 키를 변경하기 위해 사용된다. Key Change의 데이터 구조는 [그림 6]과 같다. Waiting Time은 Algorithm Change에서와 같이 새 암호화 키 사용에 대한 동기화를 위해 사용된다. Key_Num은 변경하고자 하는 암호화 키의 개수를 나타낸다. Key_Index t는 암호화 키를 바꿀 암호화 알고리즘에 대한 인덱스 값, Key_Length t는 Key t필드의 길이 (1<= t <= n, n = Key_Num), Key는 변경할 키이다. Key Change는 좀 더 강력한 암호화를 위해 주기적으로 전송된다. 전송되는 주기는 SCPR을 사용하는 애플리케이션에서 결정한다.



[그림 6] Key Change의 데이터 구조
[Fig. 6] Data Structure of Key Change

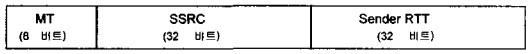
(7) Packet Ack

Packet Ack는 서버와 각 클라이언트 사이의 라운드 트립 시간을 측정하기 위해 특정 SCPR 패킷을 받은 모든 클라이언트에 의해 서버로 보내진다. Packet Ack의 데이터 구조는 그림 11과 같다. MT는 이 패킷이 어떤 패킷에 대한 응답인지를 나타내기 위해 사용되는 필드로 SCPR 헤더의 MT 중에서 Init Info(2), Algorithm Change(4)와 Key Change(5)의 값을 가질 수 있다. SSRC는 이 데이터를 보내는 클라이언트의 SSRC 식별자이다. 이 SSRC 식별자는 어떤 클라이언트가 데이터를 보냈는지 구별하고 각 클라이언트의 라운드 트립 시간을 계산하기 위해 사용된다. Sender RTT는 전에 서버가 보낸 패킷이 클라이언트에게 도착하는데 걸리는 시간으로 전 패킷을 받은 시간에서 전 패킷의 Timestamp 필드값을 뺀 값이다. 평균 라운드 트립 시간은 다음과 같이 계산한다.

$$RTT = \frac{t_1 + t_2}{2}$$

(RTT는 평균 라운드 트립 시간, t_1은 Packet Ack 패킷이 서버에 도착한 시간에서 Packet Ack 패킷의 Timestamp 필드 값을 뺀 값, t_2는 Sender RTT 필드 값)

이 라운드 트립 시간은 키 값이나 암호화 알고리즘을 바꿀 때 각 클라이언트 사이의 동기화를 위해 사용한다.



[그림 7] Packet Ack의 데이터 구조
[Fig. 7] Data Structure of Packet Ack

(8) Session Leave Request

클라이언트가 RTP 세션을 떠나기 전에 서버에 전달하는 메시지이다. SSRC 필드는 RTP 세션을 떠나려는 클라이언트의 SSRC 식별자이다. 이 패킷은 사용 시간으로 요금을 부과하는 서비스에서 요금을 부과하기 위한 목적으로 사용될 수 있다.

(9) Session Leave

서버가 특정 클라이언트가 RTP 세션에서 떠났다는 것을 다른 클라이언트들에게 알려주는 메시지이다. SSRC 필드는 RTP 세션을 떠난 클라이언트의 SSRC 식별자이다.

(10) Session End

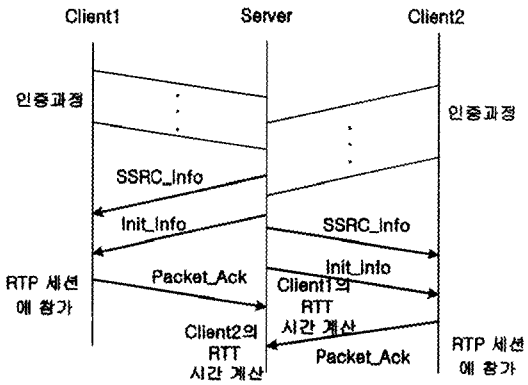
RTP 세션이 종료될 때 SCPR 서버가 현재 세션에 참여하고 있는 모든 클라이언트에게 보내는 메시지이다.

3.3 RTP 보안 제어 프로토콜의 동작 절차

(1) 클라이언트가 RTP 세션에 참여하기 전의 절차

RTP 세션에 참여하기 전에 클라이언트는 인증 프로토콜을 사용하여 인증을 받은 후, RTP 보안 제어 프로토콜의 SSRC Info와 Init Info 패킷을 RTP 보안 제어 프로토콜 서버로부터 받아 SSRC 식별자,

처음 사용할 암호화 알고리즘의 인덱스와 각 암호화 알고리즘에서 사용할 암호화 키 값을 얻는다. 클라이언트는 Init Info 패킷을 받은 후에 서버에 Packet Ack 패킷을 보내 서버가 클라이언트와의 라운드 트립 시간을 계산하도록 한다. 그 후에 클라이언트는 RTP 세션에 참여한다. 메시지 흐름은 [그림 8]과 같다.

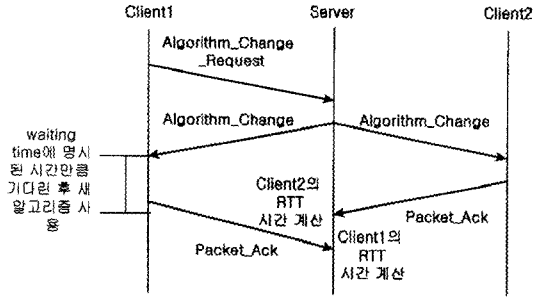


[그림 8] 클라이언트가 RTP 세션에 참여하기 전의 절차

[Fig. 8] Procedure of client's participation in RTP session

(2) 클라이언트의 암호화 알고리즘 변경 절차

암호화 알고리즘을 변경하려는 클라이언트는 서버에 Algorithm Change Request 패킷을 보내고 이 패킷을 받은 서버는 모든 클라이언트에게 Algorithm Change 패킷을 보낸다. Algorithm Change 패킷을 받은 클라이언트는 서버에 Packet Ack를 보내고 이 패킷을 받은 서버는 각 클라이언트와의 라운드 트립 시간을 계산한다. 클라이언트가 Algorithm Change 패킷을 받고 Waiting Time 만큼 기다린 후 새 암호화 알고리즘을 사용한다. 메시지 흐름은 [그림 9]와 같다.

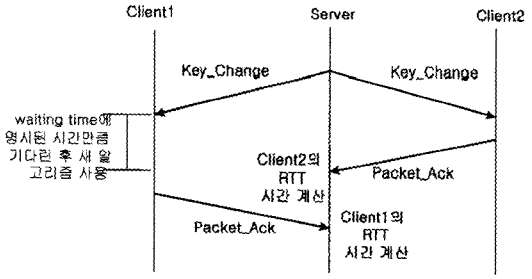


[그림 9] 클라이언트의 암호화 알고리즘 변경 절차

[Fig. 9] Procedure of Client's Algoritm Change

(3) 암호화 키 변경 절차

암호화 키를 변경하는 경우에는 서버가 모든 클라이언트에게 Key Change 패킷을 전송하고 Key Change 패킷을 받은 클라이언트는 서버에 Packet Ack를 보낸다. 이 패킷을 받은 서버는 각 클라이언트와의 라운드 트립 시간을 계산한다. 클라이언트가 Key Change 패킷을 받은 후에 Waiting Time 만큼 기다린 후 새 암호화 키를 사용한다. 메시지 흐름은 [그림 16]과 같다.



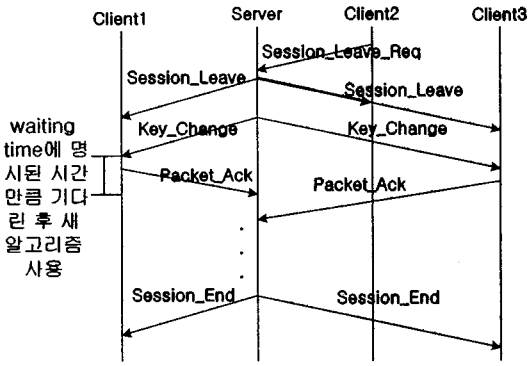
[그림 10] 암호화 키 변경 절차

[Fig. 10] Procedure of Encryption Key Change

(4) 세션 종료와 클라이언트가 세션을 떠나는 절차

RTP 세션을 떠나려는 클라이언트는 서버에 Session Leave Request 패킷을 보낸다. 이 패킷을 받은 서버는 클라이언트들에게 Session Leave 패킷을 보내 특정 클라이언트가 RTP 세션을 떠났다는 것을 알려준다. 그 후에 서버는 클라이언트들에게 Key Change 패킷을 보내 클라이언트의 암호화 키를

변경하도록 한다. 새 라운드 트립 시간을 계산하기 위해 Key Change 패킷을 받은 클라이언트들은 서버에 Packet Ack 패킷을 보낸다. 또한 RTP 세션이 끝난 경우에 서버는 세션에 참여한 모든 클라이언트들에게 Session End 패킷을 보내 RTP 세션이 끝났다는 것을 알려준다. 메시지 흐름은 [그림 17]과 같다.

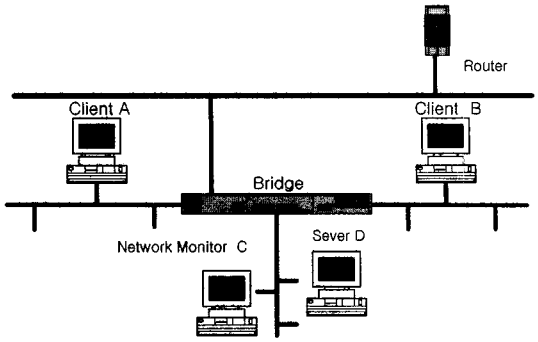


[그림 11] 세션 종료와 클라이언트가 세션을 떠나는 경우
[Fig. 11] Session End and Session Leave

4. RTP 보안 제어 프로토콜 구현 및 실험 결과

4.1 구현 환경

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



[그림 12] 실험 환경
[Fig. 12] Simulation Environments

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 RTP 보안 제어 프로토콜 서버와 클라이언트의 동작을 확인하기 위하여 TCP 패킷을 tcpdump프로그램을 사용하여 dump한 결과는 다음과 같다.

[그림 13]은 SSRC 클라이언트 B가 SSRC 서버 C에 연결하는 절차를 보인다. 단계 1은 B가 D에게 TCP 연결을 위한 request를 보내고 D가 그 요구를 받아들인 후 B에게 연결 요구에 대한 응답을 서버에 보내는 절차이다. 단계 2는 TCP 연결설정 후 D가 B에게 SSCR Info 패킷을 보내는 절차이다. 단계 3은 D가 B에 Init Info 패킷을 보내고 B는 D에게 Packet Ack을 보내는 절차이다.

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
[Fig. 13] Simulation Results of SSRC Info and Init Info

[그림 14]는 B의 요청에 의해 암호화 알고리즘을 변경하는 절차를 보인다. 단계 1은 B가 D에게 알고리즘 변경을 요청하기 위해 Algorithm Change Request를 전송하는 절차이다. 단계 2는 D가 다른 모든 클라이언트들에게 B의 알고리즘 변경을 알리기 위해 Algorithm Change를 전송하는 절차이다. 단계 3은 Algorithm Change를 받은 클라이언트들이 D에게 Packet Ack를 전송하는 절차이다.

[그림 15]는 암호화 키를 변경하는 절차를 보인다. 단계 1은 D가 모든 클라이언트들에게 암호화 키의 변경을 알리기 위해 Key Change를 전송하는 절차이다. 단계 2는 Key Change를 받은 클라이언트들이 D에게 Packet Ack를 보내는 절차이다.

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
[Fig. 14] Simulation Result of Algorithm Change Request and Algorithm Change

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
[Fig. 15] Simulation Result of Key Change

[그림 16]은 B가 세션을 떠나는 절차를 보인다. 단계 1은 세션에서 떠나려는 B가 D에게 Session Leave Request를 전송하는 절차이다. 단계 2는 D가 다른 모든 클라이언트들에게 B가 세션에서 떠났다는 것을 알려주기 위해 Session Leave를 전송하는 절차이다.

단계 3은 D가 암호화 키를 변경하기 위해 다른 클라이언트들에게 Key Change를 전송하고 이 메시지를 받은 클라이언트들이 D에게 Packet Ack를 전송, 그리고 D와 B가 TCP 연결을 해지하는 절차이다.

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
[Fig. 16] Simulation Result of Session Leave Request and Session Leave

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 RTP 보안 제어 프로토콜 서버가 다른 클라이언트에서 사용하는 암호화 키를 변경함을 확인하였다.

5. 결론

본 논문에서는 암호화 알고리즘과 암호화 키를 변경할 수 있는 RTP 보안 제어 프로토콜 을 설계 구현하였다. 클라이언트가 네트워크의 트래픽이나 부하로 인해 현재 사용하고 있는 암호화 알고리즘이 적합하지 않다고 판단하면 클라이언트는 적당한 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 후에 서버는 Key Change 메시지를 다른 클라이언트들에게 전송하여 암호화 키를 변경하도록 하였다. 따라서 멀티미디어 서비스 이용을 중지한 클라이언트로부터 서비스를 보호할 수 있었다.

실제 망에서 실험한 결과 본 논문에서 제안한 방식이 네트워크 트래픽과 컴퓨터 부하에 적응하기 위해 암호화 알고리즘을 변경하고, 암호화 키를 변경하여 RTP 데이터를 보호할 수 있음을 확인하였다.

※ 참고 문헌

[1] 이상헌, 도미선, 이재용, "인터넷에서의 멀티미디어 통신과 서비스," 한국통신학회지, 제 13권, 제 2호, pp. 65 - 81, 1996.

[2] 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, Sept. 1998.

[3] Raj Jain, "Multimedia over IP : RSVP, RTP, RTCP, RTSP," http://www.cis.ohio-state.edu/~jain/cis788-97/ip_multimedia/index.htm.

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

[5] 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.

[6] 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 om Computers and Communications*, pp. 543 - 548, Jun. 1998.

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

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

[9] 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.

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

[11] 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 전자계산학과 졸업(공학사)
1993년 인하대학교 대학원 전자계산공학과 졸업(공학석사)
2002년 인하대학교 대학원 전자계산공학과 졸업(공학박사)
1999～현재 청강문화산업대학 컴퓨터소프트웨어과 교수
관심분야 : 정보보안, QoS 라우팅, 무선통신
E-mail : jjhong@chungkang.ac.kr