情報保護學會論文誌
第13券 第6號, 2003. 12

멀티캐스트 환경에서의 패킷 손실을 고려한 인증기법 설계 및 분석

임 정 미*†, 박 철 훈*, 유 선 영*, 박 창 섭*‡

A Design and Analysis of Authentication Scheme for Tolerating Packet Loss in the Multicast Environment

Jeung-Mi Rim*†, Chol-Hoon Park*, Sun-Yong You*, Chang-Seop Park*‡

요 약

본 논문에서는 인터넷 멀티캐스트 환경하에서의 멀티미디어 스트리밍 데이터에 대한 인증 메커니즘을 제안한다. 최근, 신뢰성 있는 멀티캐스트 전송에 응용되어지고 있는 패킷 계층에서의 전방 오류수정부호를 멀티캐스트 인증 메커니즘과 연동시킨다. 이와 관련하여, 전송과정에서 일부 패킷들이 손실되어진다고 할지라도 다른 패킷들에 대한 개별 인증에는 영향을 미치지 않게 하기 위하여 Reed-Solomon 삭제부호를 적용한다.

ABSTRACT

Proposed in this paper is an authentication mechanism for multimedia streaming data in the Internet multicast environment. The multicast authentication mechanism is coupled with the packet-level forward error correction code which has been recently applied for a reliable multicast transport transmission. Associated with this, Reed-Solomon erasure code is chosen for tolerating packet loss so that each of the received packets can be authenticated independently of the lost packets.

keyword :

Ⅰ. 서 론

인터넷 사용자가 늘어나고, 망 대역폭이 커짐에 따라 인터넷을 통하여 오디오나 비디오와 같은 멀티미디어 스트리밍 데이터(multimedia streaming data)를 제공하는 서비스가 늘고 있다. 스트림은 길이가 매우 긴 비트열로, 전체 스트림을 수신한 후에 인증을 하게 될 경우, 과도한 지연이 발생하므로 적합하지 않고, 블록이나 패킷의 단위로 처리를 하는 것이 더 효율적이다. 스트리밍 서비스는 VOD와 같이 송신자가 서비스할 스트림 전체를 이미 저장하고 있는 off-line 방식과, 생방송과 같이 실시간으로 서비스되는 on-line 방식으로 구분된다.[1] 인터넷과 같은 개방형 네트워크 상에서, 안전한 데이터 전송을 위한 데이터의 기밀성 및 인증 서비스를 제공하기 위한 기반 연구는 이전에는 유니캐스트 환경에서 활발하였으나, 최근에는 규모가 큰 수신자 그룹에게 전달되는 데이터에 대한 근원지 인증 및 무결성 서비스를 제공하는 멀티캐스 인증기법에 대한 연구가 활발히 진행되고 있다. 멀티캐스트 인증이외의 멀티캐스트 보안에 대

* 이 연구는 2003학년도 단국대학교 대학연구비의 지원으로 연구되었습니다.
** 단국대학교 전자계산학과({redpig3, csp0}@dankook.ac.kr)
† 주저자, ‡ 교신저자, 논문접수일 : 2003년 10월 14일, 심사완료일 : 2003년 12월 1일

한 기존 연구는 주로 데이터에 대한 기밀성 보장을 위한 암호화 키, 즉 그룹키에 대한 분배 및 갱신에 대한 연구에 집중되어왔다. 실제로 암호화 자체만으로도 어느 정도의 무결성 서비스 제공이 가능하지만, 응용환경에 따라 기밀성 서비스가 요구되지 않는 경우도 있고, 특히 다수의 수신자 그룹이 동일한 그룹키를 공유하는 환경에서는 암호화 자체만으로는 데이터에 대한 무결성과 송신자에 대한 확인, 즉 인증이 불가능하게 된다. 멀티캐스트 인증에서의 요구사항은 일종의 부인봉쇄 기능이 내재되어 있어야 한다.

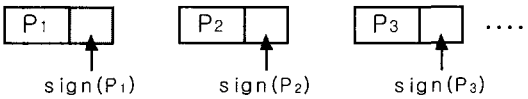
멀티캐스트 스트림의 무결성을 확인하기 위한 가장 직접적인 접근 방식은 메시지에 메시지 인증코드(MAC : Message Authentication Code)를 첨부하는 MAC 기반의 인증방식이다. MAC 방식은 송신자와 수신자가 비밀키 k를 공유하고, 스트림 $M=M_1M_2\cdots$를 구성하는 각 패킷 M_i에 대해서 MAC 값, $MAC_k(M_i)$을 해당 패킷에 첨부하는 방식이다. 이 방식은 송수신자가 일대일로 대응되는 유니캐스트 환경에서는 가능하나, 하나의 송신자와 규모가 큰 수신자 그룹으로 구성되는 멀티캐스트 환경에서는 송신자와 그룹 구성원 모두가 동일한 키를 공유해야 하므로 키 값을 가지고 있는 누구든지 MAC 값을 위조할 수 있어 근원지(송신자) 인증 서비스를 제공할 수 없다는 측면에서 멀티캐스트 환경에서는 적합하지 않다. MAC 이외의 접근 방식으로는 개별 서명방식과 체인방식이 있다. 첫째, 개별 서명방식은 각 패킷마다 서명을 첨가하는 방식으로 매 블록마다 서명 작업을 해야 함으로 높은 계산량과 서명 첨가에 따른 오버헤드가 발생한다. 둘째, 체인방식은 매 패킷마다 다른 패킷들의 해쉬 값을 첨가하고, 서명은 특정 한 개의 패킷에만 하는 방식이기 때문에 계산량이 낮고, 오버헤드도 적어 효율적이지만, 블록 내의 한 개의 패킷이라도 전송과정에서 손실이 된다면 다른 패킷들에 대한 인증을 수행할 수 없다는 단점을 가지고 있다. 본 논문에서는 블록 내의 특정 패킷들이 전송 도중 손실이 된다고 할지라도 수신된 다른 패킷들에 대한 인증을 가능하게 하는 오류수정부호에 기반을 둔 기법을 제안한다.

본 논문의 2장에서는 패킷 손실을 허용하는 멀티캐스트 인증 기법들에 대한 기존 제안들을 소개하고, 3장에서는 패킷 손실에 대처하기 위한 FEC(Forward Error Correction)을 소개한다. 4장에서는 3장에서 소개한 FEC 기법 중, RSE(Reed-Solomon Erasure)를 이용한 신뢰성 있는 멀티캐스트 인증 기법을 설계하고, 5장에서는 2장에서 소개한 이전의 인증 시스템과 4장에서 설계한 인증 시스템을 비교 분석한다.

Ⅱ. 관련 연구

2.1 개별 서명 방식

개별 서명 방식은 [그림 1]과 같이, 스트림을 여러 개의 패킷들로 나누고, 각 패킷 당 그 패킷의 해쉬 값에 대한 서명을 첨부한다. 이 방식은 패킷 단위별로 인증이 가능하고, 송, 수신자간의 패킷처리 버퍼는 1개만 필요하다. 특정 패킷이 손실되었을 경우에도, 다른 패킷들에 대한 인증에는 영향을 미치지 않는다는 장점을 가진다. 하지만 개별 서명방식은 매 패킷마다 서명 작업을 해야 함으로 높은 계산량과 오버헤드가 발생한다.

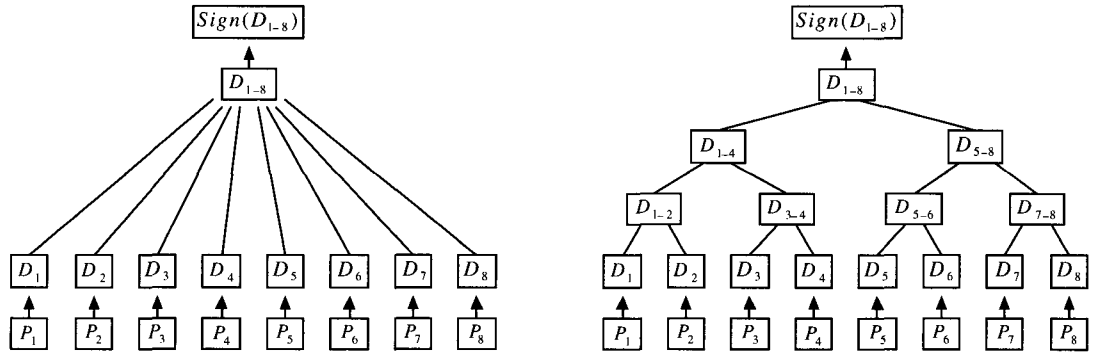


〔그림 1〕 개별 서명 방식

2.2 패킷 체인 방식

패킷 체인 방식은 패킷마다 서명을 하는 개별 서명방식의 단점을 보완하고자 스트림을 블록으로 나누고, 한 개의 블록은 n개의 패킷들로 구성하여 각 패킷에 대하여 그 패킷이 존재하는 블록내의 다른 모든 패킷들의 해쉬 값과 서명 값을 첨가한다. 대표적인 체인 방식으로 star 체인 방식과 tree 체인 방식이 있다.[2,3]

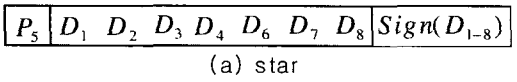
star 체인 방식은 [그림 2-a]와 같이 각 패킷들 (P_1, P_2, ... , P_n)에 대해서 해쉬함수 $H(\ )$를 적용한 해쉬 값(D_1, D_2, ⋯, D_n)을 leaf 노드로 하고, 합쳐진 leaf 노드들의 해쉬 값(D_{1-8})에 서명을 한 $sign(D_{1-n})$을 root 노드로 한다. 이때의 root 노드를 블록서명이라 한다. 송신자는 보내고자하는 패킷의 블록내의 위치, 블록내의 다른 패킷들의 해쉬 값, 블록서명을 인증 정보로 하여 해당 패킷에 첨가하여 보낸다. 처음 도착하는 패킷에 대한 인증이 확인되면 첨부된 다른 패킷들의 해쉬 값을 저장해놓고, 다음 번 패킷부터는 이미 저장된 해쉬 값 중 해당 패킷의 해쉬 값을 계산하고 비교하게 됨으로, [그림 3-a]에서 보는 바와 같이 두 번째 패킷부터는 다른 패킷들의 해쉬 값과, 블록서명은 오버헤드가 된다.



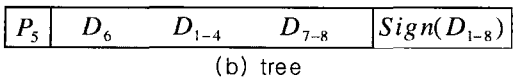
〔그림 2〕 star 체인 방식과, tree 체인 방식

tree 체인 방식은 [그림 2-b]와 같이 n개의 패킷들 ($P_1, P_2, \cdots, P_n$)의 각 해쉬 값($D_1, D_2, \cdots, D_n$)을 leaf 노드로 하고, 두 개의 leaf 노드들에 대한 해쉬 값을 parent 노드($H(D_1 \| D_2) = D_{1-2}$, $H(D_3 \| D_4) = D_{3-4}$, ... , $H(D_{1-2} \| D_{3-4}) = D_{1-4}$, ...)로 하는 작업을 반복하여 D_{1-n}를 구한 후, 이 값에 서명을 한 $sign(D_{1-n})$을 root 노드로 하는 2진 트리에 기반을 두고 있다. tree 체인 방식의 송신자는 보내고자 하는 패킷의 블록내의 위치, 루트까지의 경로에 존재하는 sibling 노드, 블록서명을 인증 정보로 하여 해당 패킷에 첨가하여 보낸다. 첨가된 root까지의 경로에 존재하는 sibling 노드를 이용하여 처음 도착하는 패킷을 인증하고, 인증이 되면 sibling 노드들을 저장한 후, 다음 번부터는 root 노드까지 계산하여 비교하지 않고, 저장된 sibling 노드들 중에 있으면 인증이 이루어진다. star 방식보다는 적지만 역시 tree 방식에서도 그림 3-b와 같이 적지 않은 오버해드가 발생한다.

start 방식과 tree 방식 모두 패킷마다 해당 패킷에 대한 인증정보가 존재하므로 특정 패킷의 손실이 다른 패킷에 대한 인증을 하는 데에는 영향을 주지 않는다. 그러나 긴 스트림을 전송 할 때 chaining 작업으로 인한 오버해드는 네트워크의 트래픽을 가중시키며, 블록 단위로 처리되기 때문에 지연(delay)이 발생된다.



(a) star



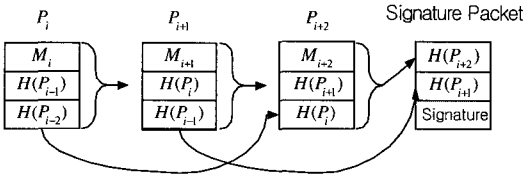
(b) tree

〔그림 3〕 star 체인 방식과 tree 체인 방식에서의 패킷 구성

2.3 선별적 체인 방식

EMSS(Efficient Multi-chained Stream Signature)[4]란 이름으로 제안된 선별적 체인방식 역시 star 방식과 tree 방식과 같이 스트림을 블록으로 나누고 한 개의 블록은 n개의 패킷들로 구성한다. [그림 4]와 같이 개별 패킷에 선별적으로 1개 이상의 다른 패킷에 대한 해쉬 값을 첨부하고 마지막 한 개의 패킷에만 서명 값을 첨부한다. 그림에서 보는 바와 같이 서명 패킷(Signature Packet)의 서명필드에는 $H(P_{i+1})$와 $H(P_{i+2})$에 대한 서명 값이 저장된다.

이 방식은 예를 들어, $i+1$번째의 패킷이 손실되었을 경우, P_{i+1}에 첨부된 $H(P_i)$와 $H(P_{i-1})$은 P_i와 P_{i+2}에도 들어있으므로 복구가 가능하다. 즉, 패킷이 손실되었을 경우에 손실된 패킷의 해쉬 값이 다른 패킷에도 존재하고 있으므로 인증이 가능하게 된다.



〔그림 4〕 선별적 체인 방식

Ⅲ. 멀티캐스트 채널과 오류수정 부호

인터넷과 같이 규모가 큰 네트워크 상에서 전송되는 패킷들은 통신오류나 라우터에서의 혼잡 등으로 인하여 손실될 가능성이 매우 높다. 이런 패킷 손실을 해결하기 위해서는 기본적으로 ARQ(Automatic

Retransmission reQuest)기법이 사용된다. 유니캐스트 환경에서는 손실된 패킷을 재전송 시켜주는 ARQ기법이 적합하지만, 수천 혹은 수만 명 이상의 수신 그룹이 존재하는 멀티캐스트 환경 하에서 ARQ를 사용시, 손실된 패킷에 대한 NAK(Negative Acknowledgment) 패킷의 개수가 증대되고, 해당 손실된 패킷에 대한 재전송이 수행 되어야하기 때문에 이로 인한 네트워크의 혼잡과 송신자 측 서버의 부하는 증대된다. 최근 이를 보완하기 위한 SRM(Scalable Reliable Multicast)[5]과 RMTP(Reliable Multicast Transport Protocol)[6] 등과 같이 신뢰성 있는 멀티캐스트 전송 프로토콜이 제안되고 있지만, 복잡하고 아직 표준화 작업이 완성되어있지 않다. 특히, 응용 환경마다의 요구 사항이 상이하기 때문에, 단일 멀티캐스트 전송 프로토콜을 다양한 응용 분야마다 일률적으로 사용하는 데에는 무리가 따른다. 최근에는 신뢰성이 요구되는 멀티캐스트 환경에 적합한 오류 수정부호에 기반을 둔 FEC에 대한 연구가 진행되고 있다. 오류수정부호는 비트단위의 채널오류를 제어하기 위한 것이었으나 최근, 패킷의 손실을 오류수정부호에 있어서 삭제(erasure)로 간주하여, 패킷단위의 오류를 제어하기 위한 방안으로 응용 되고 있다. 잘 알려진 삭제부호(erasure codes)로는 Reed-Solomon 부호[7]와 Tornado 부호[8]가 있다.

FEC 기법에서의 삭제 부호화(erasure encoding) 방식은 원본 데이터와 더불어 이것의 일부가 손실되었을 경우에 손실된 부분을 복구하는데 소요되는 패리티 데이터를 블록으로 구성하여 전송하게 된다. 본 제안에서 사용 될 Reed-Solomon Erasure(RSE) 부호는 이 패리티 데이터를 생성하는데 이용된다. RSE 부호는 길이가 l 비트인 k개의 데이터 symbol $d_1, d_2, \cdots, d_k$와 $n-k$개의 패리티 symbol $p_1, p_2, \cdots, p_{n-k}$로 구성된다. 데이터 symbol $d_1, d_2, \cdots, d_k$는 Galois Field GF(2^l)의 구성 원소이다. α를 GF(2^l)의 원시 원소라고 가정하면, 패리티 symbol $p_1, p_2, \cdots, p_{n-k}$는 식(1)을 이용하여 다음과 같이 계산된다.

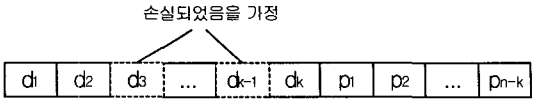
$$p_j = F(\alpha^{j-1}), \qquad (1)$$

where

$F(x) = d_1 + d_2x^1 + \cdots + d_kx^{k-1}$

and $j=1,2,\ldots,n-k$

이와 같은 부호를 (n, k) 부호, 즉 k개의 데이터 symbol에 $n-k$개의 symbol이 첨가되어 n개의 symbol로 구성된 블록 부호(block code)로 정의한다.



〔그림 5〕 패킷 손실의 예

수신자 측에서는 k개의 데이터 symbol을 모두 받았다면 복호화 과정이 생략되지만, 그렇지 않을 경우에는 복호화 과정을 거치게 된다. 즉, 손실된 데이터 symbol의 개수가 패리티 symbol의 개수 $n-k$보다 작을 경우에 한해서는 다음과 같은 복호화 과정을 통해서 손실된 데이터 symbol을 복구할 수가 있게 된다. 예를 들어, [그림 5]에서와 같이 패킷 d_3와 d_{k-1}이 수신되지 않았다고 가정하자. 수신측에서는 식(1)을 이용하여 $d_3=x_1$ 그리고 $d_{k-1}=x_2$로 하는 다음과 같은 연립방정식을 계산하여 손실된 패킷을 복구할 수 있다. 이 연립방정식의 해(solution)가 항상 존재함에 대한 논의는 다음 장에서 하기로 한다.

$$p_1 = F(\alpha^0) = d_1 + d_2(\alpha^0)^1 + x_1(\alpha^0)^2 + \cdots + x_2(\alpha^0)^{k-2} + d_k(\alpha^0)^{k-1}$$

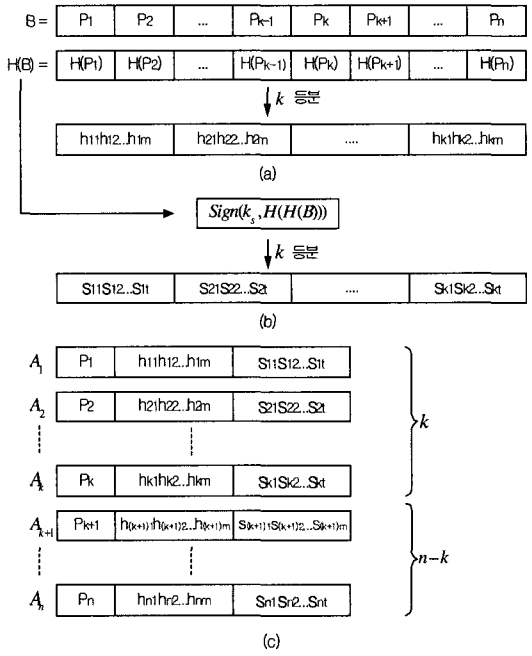
$$p_2 = F(\alpha^1) = d_1 + d_2(\alpha^1)^1 + x_1(\alpha^1)^2 + \cdots + x_2(\alpha^1)^{k-2} + d_k(\alpha^1)^{k-1}$$

Ⅳ. RSE에 기반을 둔 멀티캐스트 인증 기법 제안

이번 장에서는 패킷이 손실되는 멀티캐스팅 환경하에서, 패킷들에 대한 인증을 효과적으로 수행할 수 있는 기법을 RSE 부호를 기반으로 제안한다. 매 패킷마다 해쉬 값과 서명 값을 첨부하는 대신에 패킷 블록별로 블록 전체에 대한 해쉬 값과 서명 값을 RSE 부호화화고 그 결과를 블록내의 각 패킷들에 나누어 첨부한다. 여기서 적용되는 RSE 부호는 손실된 패킷 전체를 복구하는 것이 아니라, 나머지 수신된 패킷들에 대한 인증을 성공적으로 수행하기 위하여, 손실된 패킷들에 대한 해쉬 값 및 서명 값만을 복원하기 위한 목적으로 사용된다.

4.1. 패킷 블록 부호화

스트림을 여러 개의 블록들로 나누고, 다시 각각



[그림 6] 패킷 블록 부호화

의 블록 B는 n개의 패킷들($P_1 \| P_2 \| \cdots \| P_n$)로 구성된다. n개의 패킷 중에서 k개의 패킷만 수신이 되어도 그 수신된 패킷들에 대한 개별적인 인증을 가능하게 하기 위해서 (n, k) RSE 부호를 적용한다. 패킷 블록 부호화는 모든 블록에 대해서 동일하게 적용이 되기 때문에, 본 논의에서는 한 개의 블록을 기준으로 설명한다.

[그림 6-(a)]와 같이 블록 B를 구성하는 n개의 패킷 $P_1, P_2, \cdots, P_n$에 대한 각각의 해쉬 값 $H(P_1), H(P_2), \cdots, H(P_n)$을 계산하여 식(2)에서와 같이 $H(B)$를 구성한다.

$$H(B) = H(P_1) \| H(P_2) \| \cdots \| H(P_n) \tag{2}$$

$|H(B)| = N$(바이트)로 가정하고, $H(B)$를 k등분하여 $m(=N/k)$ 바이트로 이루어 진 k개의 그룹 $(h_{11}, h_{12}, \cdots, h_{1m})$, $(h_{21}, h_{22}, \cdots, h_{2m})$, $\cdots$, $(h_{k1}, h_{k2}, \cdots, h_{km})$을 구성한다. 이를 기반으로 식(3)과 같이 GF (2^8)상의 (n, k) RSE 부호를 적용하여 다음과 같은 $n-k$개의 패리티 그룹을 생성한다. 이때, α는 GF (2^8)의 원시원소이다.

$$(h_{(k+1)1}, h_{(k+1)2}, \cdots, h_{(k+1)m}),$$
$$(h_{(k+2)1}, h_{(k+2)2}, \cdots, h_{(k+2)m}),$$
$$(h_{n1}, h_{n2}, \cdots, h_{nm})$$

$$F_j(\alpha^0) = h_{(k+1)j}, \quad F_j(\alpha^1) = h_{(k+2)j}, \quad \cdots,$$
$$F_j(\alpha^{n-k-1}) = h_{nj}, \tag{3}$$

where

$$F_j(x) = h_{1j} + h_{2j} \cdot x^1 + h_{3j} \cdot x^2 + \cdots + h_{kj} \cdot x^{k-1} \quad \text{and} \quad j = 1, \ldots, m.$$

이와 같이 생성된 n개의 그룹을 [그림 6-(c)]와 같이 각각 n개의 패킷에 첨부한다. 또한, 해쉬 값 $H(B)$에 대해서 다시 한번 해쉬 함수를 적용한 결과 값 $H(H(B))$를 송신자의 서명용 키 k_s로 서명하여 $sign(k_s, H(H(B)))$을 생성한다. 이 서명 값도 위에서 논의했던 해쉬 값의 경우와 마찬가지로 k등분한다. 즉, 서명 값의 길이를 M 바이트로 가정할 때, $M / k = t$ 이면 $(s_{11}, s_{12}, \cdots, s_{1t})$ $(s_{21}, s_{22}, \cdots, s_{2t}) \cdots (s_{k1}, s_{k2}, \cdots, s_{kt})$와 같은 t 개의 바이트로 이루어 진 k 개의 그룹을 구성할 수 있다. 다시 여기에 식(4)와 같이 (n, k) RSE 부호를 재 적용하여 $n-k$개의 패리티 그룹을 생성한다.

$$(s_{(k+1)1}, s_{(k+1)2}, \cdots, s_{(k+1)t}),$$
$$(s_{(k+2)1}, s_{(k+2)2}, \cdots, s_{(k+2)t}),$$
$$(s_{n1}, s_{n2}, \cdots, s_{nt})$$

$$G_j(\alpha^0) = s_{(k+1)j}, \quad G_j(\alpha^1) = s_{(k+2)j}, \quad \cdots,$$
$$G_j(\alpha^{n-k-1}) = s_{nj} \tag{4}$$

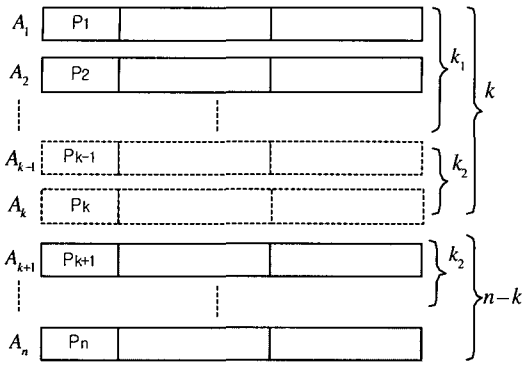
where

$$G_j(x) = s_{1j} + s_{2j} \cdot x^1 + s_{3j} \cdot x^2 + \cdots + s_{kj} \cdot x^{k-1}$$
and $j = 1, \ldots, t$.

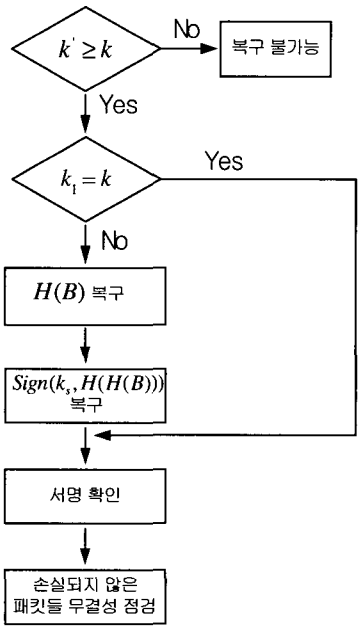
마찬가지로 이 n개의 그룹은 각각 n개의 패킷에 첨부되어, $A_1, A_2, \cdots, A_n$의 인증 패킷(authenticated packet)이 구성되며, 최종적으로 이 인증 패킷들이 멀티캐스팅 되어진다.

4.2 패킷 블록 복호화

[그림 7]에서와 같이 송신된 n개의 인증 패킷들 $\{A_1, A_2, \cdots, A_n\}$ 중에서 $k'(=k_1+k_2)$개의 인증 패



〔그림 7〕 패킷 블록 복호화



〔그림 8〕 복호화 순서도

킷들이 수신이 되었다고 가정하자. (즉, $\{A_1, A_2, \cdots, A_k\}$ 중에서 k_1개, $\{A_{k+1}, A_{k+2}, \cdots, A_n\}$중에서 k_2개). 패킷들은 전송도중 무작위로 손실(random loss)이 되지만 논의의 편의상 일반성을 잃지 않고, $\{A_1, A_2, \cdots, A_{k_1}\}$와 $\{A_{k+1}, A_{k+2}, \cdots, A_{k+k_2}\}$이 수신되었다고 가정하자. 만약, $k' \geq k$일 경우에는 해당 블록에 대한 성공적인 복호화가 가능하기 때문에 수신된 인증 패킷들에 대해서는 개별적인 인증이 가능하게 된다. 특히, $k_1 = k$일 경우는 패킷 블록에 대한 복호화 과정에서의 해쉬 값 및 서명 값 복구 작업은 생략된다. 그 이유는 $\{A_1, A_2, \cdots, A_k\}$ 에는 해당 블록 전체에 대한 해쉬 값 및 서명 값이 이미 포함되어있기 때문이다. 따라서, 이 경우에는 [그림 8]에서와 같이 서명에 대한 확인작업만 수행하면 된다. 하지만, $k_1 < k$의 경우 즉, $\{A_1, A_2, \cdots, A_k\}$ 중에서 1개 이상의 인증 패킷이 손실이 될 경우에는 블록 복호화 과정을 통해서 $H(B)$와 $Sign(K_s, H(H(B)))$을 복구하여야 한다.

$k_1 < k$일 경우의 해쉬 값 $H(B)$ 복구는, 결국 손실된 패킷 $\{A_{k_1+1}, A_{k_1+2}, \cdots, A_k\}$에 포함되어 있던 부분적인 해쉬 값을 복구하는 것이다. 이를 위해서 수신된 $\{A_{k+1}, A_{k+2}, \cdots, A_{k+k_2}\}$에 포함되어 있는 해쉬 값이 이용된다. 하지만, 이중에서도 k_2개 전부가 필요한게 아니라, $k - k_1 (\leq k_2)$개만 이용하면 된다. $j = 1, 2, \ldots, m$에 대해서, 식(5)에 나타나 있는 ①의 h_{ij} 부분은 손실되지 않았으므로 알고 있는 k_1개의 값이고, ②의 h_{ij} 부분은 손실되어서 수신자가 알지 못하기 때문에, 복구되어야하는 $k - k_1$개의 값이다.

$$F_j(x) = h_{1j} + h_{2j} \cdot x^1 + h_{3j} \cdot x^2 + \cdots \underset{①}{h_{k_1 j} \cdot x^{k_1 - 1}} + \underset{②}{h_{(k_1+1)j} \cdot x^{k_1} + \cdots + h_{kj} \cdot x^{k-1}} \quad (5)$$

식(5)를, 값을 알고 있는 h_{ij} 부분과 복구되어야 할 h_{ij} 부분으로 구분하면 다음과 같다.

$$F'_j(x) = F_j(x) - \left(h_{1j} + h_{2j} \cdot x^1 + h_{3j} \cdot x^2 + \cdots h_{k_1 j} \cdot x^{k_1 - 1}\right) = h_{(k_1+1)j} \cdot x^{k_1} + \cdots + h_{kj} \cdot x^{k-1}$$

$F'_j(x)$를 기반으로 다음과 같은 $k - k_1$개의 식을 계산한다.

$$\begin{aligned} F'_j(\alpha^0) &= h_{(k_1+1)j} \cdot (\alpha^0)^{k_1} + h_{(k_1+2)j} \cdot (\alpha^0)^{k_1+1} + \cdots + h_{kj} \cdot (\alpha^0)^{k-1} \\ F'_j(\alpha^1) &= h_{(k_1+1)j} \cdot (\alpha^1)^{k_1} + h_{(k_1+2)j} \cdot (\alpha^1)^{k_1+1} + \cdots + h_{kj} \cdot (\alpha^1)^{k-1} \\ &\vdots \end{aligned}$$

$$F'_j(\alpha^{k-k_1-1}) = h_{(k_1+1)j} \cdot (\alpha^{k-k_1-1})^{k_1} + h_{(k_1+2)j} \cdot (\alpha^{k-k_1-1})^{k_1+1} + \cdots \cdots + h_{kj} \cdot (\alpha^{k-k_1-1})^{k-1}$$

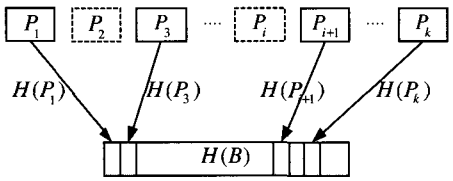
위의 식을 행렬로 표현하면 식(6)과 같고, 이때 행렬 A는 Vandermonde 행렬이기 때문에 역행렬이 존재하며 따라서, 해(solution)가 유일하게 존재한다. 따라서 손실된 인증 패킷들 $\{A_{k_1+1}, A_{k_1+2}, \cdots, A_k\}$에 포함된 부분 해쉬 값 $\{h_{(k_1+1)j}, h_{(k_1+2)j}, \ldots, h_{kj}\}$을 복구할 수가 있다.

$$A \cdot \begin{bmatrix} h_{(k_1+1)j} \\ h_{(k_1+2)j} \\ \vdots \\ h_{kj} \end{bmatrix} = \begin{bmatrix} F'_j(\alpha^0) \\ F'_j(\alpha^1) \\ \vdots \\ F'_j(\alpha^{k-k_1-1}) \end{bmatrix}, \quad (6)$$

where A =

$$\begin{bmatrix} (\alpha^0)^{k_1} & (\alpha^0)^{k_1+1} & \cdots & (\alpha^0)^{k-1} \\ (\alpha^1)^{k_1} & (\alpha^1)^{k_1+1} & \cdots & (\alpha^1)^{k-1} \\ \vdots & \vdots & \ddots & \vdots \\ (\alpha^{k-k_1-1})^{k_1} & (\alpha^{k-k_1-1})^{k_1+1} & \cdots & (\alpha^{k-k_1-1})^{k-1} \end{bmatrix}$$

궁극적으로 $H(B) = (h_{11}h_{12}h_{13}\cdots h_{1m}) \| (h_{21}h_{22}h_{23}\cdots h_{2m}) \| \cdots \| (h_{k1}h_{k2}h_{k3}\cdots h_{km})$을 구할 수 있다. 복호화 과정의 다음은 손실된 인증 패킷에 포함된 부분 서명 값을 복구해야 하는데, 이 경우는 해쉬 값을 복구하는 과정과 동일하기 때문에 생략하기로 한다. 결국, $Sign(k_s, H(H(B)))$에 해당하는 $(s_{11}s_{12}\cdots s_{1t}) \| (s_{21}s_{22}\cdots s_{2t}) \| \cdots \| (s_{k1}s_{k2}\cdots s_{kt})$도 획득할 수 있게 된다. 다음은 복구된 해쉬 값, 서명 값 그리고 서명 확인용 키를 이용하여 서명확인 작업을 수행한다. 복호화의 마지막 작업으로, 손실되지 않은 인증 패킷에 대한 무결성 점검을 수행한다. 이 과정이 필요한 이유는, 이전 단계에서 수행된 서명확인 작업은 단지 블록내에 첨부된 해쉬 값을 기반으로 행해졌기 때문이다. 따라서, 인증 패킷 A_i 내에 존재하는 패킷 P_i에 대한 무결성 점검이 요구된다. 이는 [그림 9]에서



〔그림 9〕 패킷의 무결성 점검

와 같이 수신된 패킷들에 대한 해쉬 값을 계산하고, 그 값이 $H(B)$에 존재하는지를 확인한다.

V. 제안 방식의 평가 및 비교

[표 1]에서는 2장에서 언급했던 기존 방식과의 성능비교를 보여주고 있다. 평가항목으로는 블록 당 해쉬 계산 회수, 패킷 당 첨부되는 해쉬 값 길이, 수신된 패킷을 인증하기 위해서 허용되는 최대 손실 패킷 수, 그리고 패킷 처리를 위한 수신측에서의 지연, 즉 블록의 최초 패킷을 인증하기 위해서 필요한 패킷 버퍼 수를 대상으로 하였다. 디지털 서명 항목은 4개의 방식 모두 한번만 수행되기 때문에 비교에서 제외하였고, 다만 서명확인 작업은 단순체인 방식에서는 블록을 구성하는 패킷만큼, 즉 n번이 요구되고 나머지 방식들에서는 각각 1번이 소요된다.

$|H|$ = 바이트 단위의 해쉬 값 길이
n = 블록 당 패킷 수
$n-k$ = RSE 부호에서의 패러티 블록 수.
w = EMSS에서의 패킷 당 해쉬 값 개수.

단순체인 방식에서는 블록내의 각각의 패킷들은 다른 패킷들과는 독립적으로 패킷 인증이 이루어지기 때문에 최대 손실 패킷 수에는 제한이 없다. EMSS 선별체인 방식은 n개의 패킷으로 구성된 블록을 대상으로 한 개의 패킷 당 w개의 다른 패킷에 대한 해쉬 값이 첨가될 경우에는 최대 w개의 연속된 패킷 손실(a burst loss of packets)을 허용한다. 이와 관련하여, 그래프 이론에 기반을 둔 확장된 EMSS 선별체

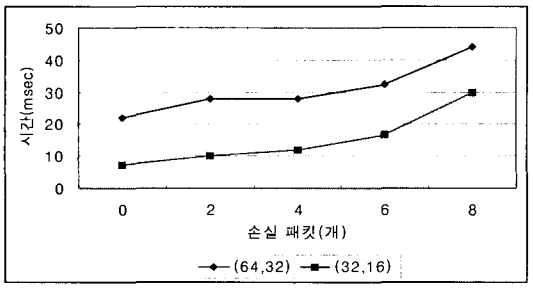
표 1. 4가지 방식의 비교 분석

	hash 및 추가연산	overhead / packet	max. packet loss	delay
단순체인 (Star)	$n+1$	$(n-1) \cdot \|H\|$	any	1
단순체인 (Tree)	$2n-1$	$(\log_2 n) \cdot \|H\|$	any	1
선별체인 (EMSS)	n+1	$w \cdot \|H\|$	a burst of w	n
제안방식 (RSE)	$n+1$+(복호연산)	$(n \cdot \|H\|) / k$	$n-k$	$k \sim n$

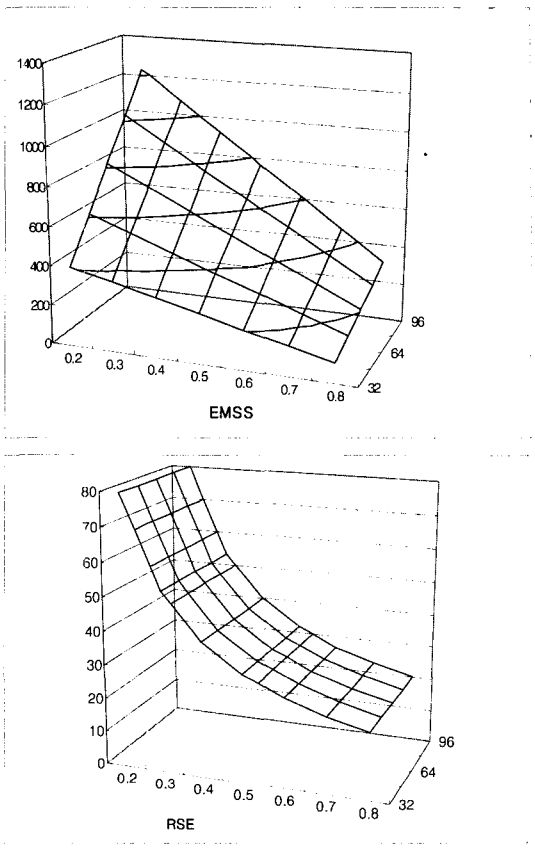
인 방식들에 대한 제안[9,10]에서는 무작위 손실(random loss) 및 연속 손실(burst loss)에 따른 수신된 패킷들에 대한 인증 성공률 분석이 행해진바 있다. 제안방식(RSE)의 경우, 수신측에서의 지연은 블록을 구성하는 최초 k개의 패킷이 성공적으로 수신이 되면 첫 패킷에 대한 인증작업이 가능해 진다. 따라서, 지연은 최소 k, 최대 n이 된다.

제안방식에서는 도표에 나와 있는 것처럼 RSE 부호에 대한 복호화 작업이 추가로 수행된다. 복호화 과정에서 가장 큰 계산량을 수반하는 곳은 행렬 A의 역행렬을 구하는 부분(식(6) 참조)이다. 복호화 과정에서 언급한 것처럼, 전송된 n개의 패킷 중에서 패킷 손실이 뒤쪽의 $n-k$개의 패킷들에 발생된다면 복호화 작업은 필요 없게 된다. 또한, 여러 개의 패킷 손실이 발생된 경우에 행렬 A의 크기에 영향을 미치는 요인은 n개의 패킷 중에서 앞쪽의 k개의 패킷들에 발생된 개수이다. 따라서, 최악의 경우는 패킷 손실이 앞쪽 k개의 패킷 모두에 발생되는 경우이다. 따라서, (n, k) RSE 부호의 복호화에 따른 계산 복잡도는 $O(\varepsilon^a)$, $\{2 \le a \le 3,\ 0 \le \varepsilon \le n-k\}$이다. RSE 복호화에 따른 계산량을 가늠하기 위하여, 역행렬 계산에 소요되는 시간과 서명확인에 소요되는 시간을 측정 하였다. n=16 그리고 k = 8일 경우에 손실된 패킷 개수가 2, 3, 4, 5일때, 해당 역행렬을 구하는데 소요된 시간은 Pentium 4(1.9 GHz)에서 각각 4ms, 5ms, 8ms, 10ms이였다. 또한, 16바이트 해쉬 값의 1024비트 RSA 서명확인에는 가장 작은 지수 값 3을 적용시 3ms이 소요되었다.

[그림 10]은 n = 64, n = 32일 때, 그리고 패킷의 크기가 1024바이트의 경우에 수신측에서 해당 블록을 복호화하는 데에 소요되는 시간(즉, 복구 시간)을 측정하였다. 블록 당 손실되는 패킷의 수를 X축에 표시하였다. 측정을 위한 도구는 Pentium 4(1.9 GHz),



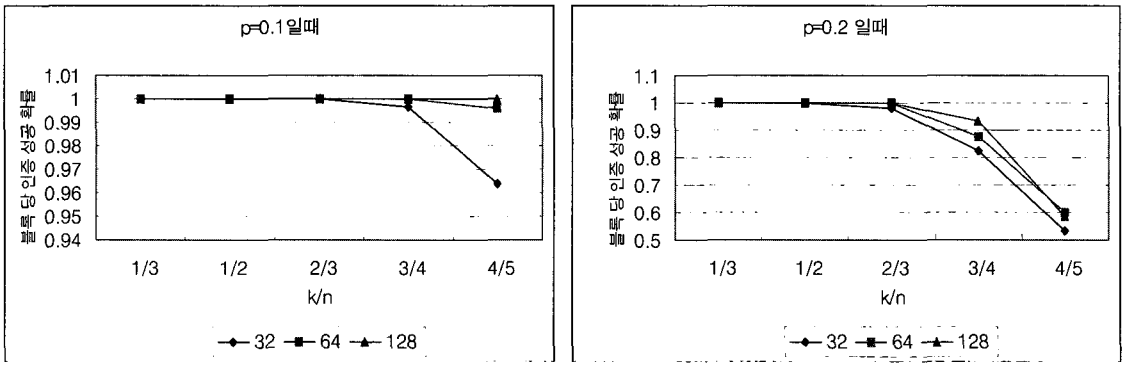
〔그림 10〕 손실된 패킷 수에 따른 복구 및 수신 측 처리 시간



〔그림 11〕 패킷 당 오버해드 비교

Windows XP 상에서 Microsoft Visual Stdio.NET으로 CryptoAPI를 기반으로 하여 작성하였으며, 해쉬 함수로는 MD5, 암호화 알고리즘은 RSA-1024를 사용하였다.

[그림 11]은 $|H|$=16바이트, n=32~96일 경우에 EMSS와 제안방식(RSE)에서의 패킷 당 오버해드(패킷 당 첨부되는 바이트 단위의 해쉬 값 크기)를 비교하였다. 비교를 위해서 동일 블록 크기 n에 대해서 동일한 개수의 연속된 패킷 손실을 허용한다고 가정하면 $w=n-k$가 된다. 그림에서는 블록 크기 n과 k/n 변화(0.2~0.8)에 따른 오버해드를 보여주고 있고, 제안방식의 오버해드가 EMSS에서 보다 상당히 우수한 성능을 보여주고 있음을 확인할 수 있다. 예를 들어, (n=32, k=8) RSE 부호가 적용될 경우에는 제안방식은 24(=$n-k$) 개의 연속된 패킷 손실을 허용한다. 따라서, EMSS의 경우 동일한 개수의 패킷 손실을 허용하기 위해서는 w=24이 된다. 제안방식의 패킷 당 오버해드는 $(n \cdot |H|)/k$=4 • 16=64바이트, EMSS의 경우는 $w \cdot |H|$=24 • 16=384바이트가 된다. EMSS의 경우



〔그림 12〕 블록 당 인증 성공 확률 비교

오버헤드가 높은 이유는 각각의 패킷에 중복되는 해쉬값들이 반복되어 첨부되기 때문이다.([그림 4] 참조)

[그림 12]에서는 (n, k) RSE 부호가 적용된 1개의 블록을 받았을 때, 블록 당 수신된 패킷들에 대한 개별적인 인증 성공확률을 n값이 32, 64, 128경우에 k/n의 비율에 따라 보여주고 있다. 패킷 당 손실 확률(packet loss probability)은 p로 표시하였다. 이 값은 실제 MBONE 측정치[11]인 10~20%로 설정하였다. 여기서의 인증 성공확률이란 무결성 점검이 성공적으로 행해진다는 의미가 아니라, 인증에 대한 확인작업을 할 수 있는 충분한 패킷들을 복구할 수 있느냐에 대한 성공확률을 의미한다. 결국 n개의 패킷 중에서 $n-k+1$ 이상의 패킷이 손실될 경우에는 인증이 실패하게 됨으로 인증 성공확률은 식(7)과 같다.

$$P_{RSE} = 1 - \sum_{i=n-k+1}^{n} \binom{n}{i} p^i (1-p)^{n-i} \qquad (7)$$

특히, 제안방식에서는 $n-k$개 이하의 연속적 패킷 손실에만 국한하는 것이 아니라 $n-k$개 이하의 무작위 패킷 손실 역시 허용한다. 그림 12에서 알 수 있듯이 한 개의 블록을 구성하고 있는 패킷의 개수가 많을수록, k/n이 작을수록 블록 당 인증 성공확률이 증가함을 알 수 있다.

Ⅵ. 결 론

본 논문에서는 인터넷 멀티캐스트 환경하에서의 멀티미디어 스트리밍 데이터에 대한 인증 메커니즘을 패킷 손실을 고려하여 제안하였다. 패킷 단위별로 인증하는 경우와 여러 패킷을 체인방식으로 그룹화하여 인증하는 경우의 오버헤드 상의 한계를 인식하여, 이에 대한 대안으로 Reed-Solomon 삭제부호를 인증 메커니즘에 연동시킨 시스템을 설계하고 이에 대한 분석을 시도하였다. 제안방식에서는 삭제부호 적용에 따른 추가 연산비용이 소요되지만 이를 적용한 결과, 허용되는 손실 패킷이 동일하다는 조건 하에서 패킷 당 오버헤드는 제안하는 방식이 기존 체인방식보다는 우수한 성능을 보여줌을 확인할 수 있었다. 멀티캐스트 인증 서비스뿐만 아니라, 멀티캐스트 환경에서의 그룹키 분배, 갱신 및 기밀성 보장위한 암호화에 있어서도 삭제부호를 이용한 신뢰성 있는 보안 서비스의 제공은 관련 상용 서비스 시스템 구축에는 필수적인 사항으로 사료된다.

참 고 문 헌

[1] R. Geranno, P. Rohatgi, "How to Sign Digital Streams", *Advances in Cryptology*-Crypto'97, LNCS 1294, pp.180~197, 1997.

[2] P. Rohatgi, "A Compact and Fast Hybrid Signature Schemes for Multicast Packet Authentication", *The 6th ACM Conference on Computer and Communications Security*, pp.93~100, 1999.

[3] C. Wong, S. Lam, "Digital Signatures for Data Flows and Multicasts", *IEEE /ACM Transactions on Networking, vol.7*, pp.502~513, 1999.

[4] A. Perrig, R. Canetti, J.D. Tygar, D. Song, "Efficient Authentication and Signing of Multicast Streams over Lossy Channels", *IEEE Symposium on Security and Privacy*, Oakland, CA, 2000.

[5] S. Floyd, V. Jacobson, C. Liu, S. McCanne, L. Zhang, "A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing", *IEEE/ACM*

Transactions on Networking, vol.5, pp.784~803, 1997.

[6] J. C. Lin, S. Paul, "RMTP: A Reliable Multicast Transport Protocol", IEEE Infocom, 1996.

[7] J. Nonnenmacher, E. Biersack, D. Towsley, "Parity-based Loss Recovery for Reliable Multicast Transmission", *IEEE/ACM Transactions on Networking, vol.6*, pp.349~361, 1998.

[8] M. Luby, M. Mitzenmacher, M.A. Shokrollahi, D. Spielman, V. Stemann, "Practical Loss-Resilient Codes, *The 29th Symposium on the Theory of Computing, ACM*, pp.150~159, 1997.

[9] P. Golle, N. Madadugu, "Authenticating Sreamed Data in the Presence of Random Packet Loss", *ISOC Network and Distributed System Security Symposium*, pp.13~22, 2001.

[10] S. Miner, J. Staddon, "Graph-based Authentication of Digital Streams", *IEEE Symposium on Security and Privacy*, Oakland, CA, 2001.

[11] M. Yajnik, S. Moon, J. Kurose, D. Towley, "Measurement and Modelling of the Temporal Dependence in Packet Loss", *IEEE Grobal Internet*, London, UK, 1996.

〈著者紹介〉

임 정 미 (Jeong-Mi Lim) 준회원
2000년 2월 : 단국대학교 전자계산학과 졸업 학사
2002년 2월 : 단국대학교 전자계산학과 석사
2002년 3월~현재 : 단국대학교 전자계산학과 박사과정
<관심분야> 정보보호, 네트워크 보안

박 철 훈 (Chol- Hoon Park) 준회원
2002년 2월 : 단국대학교 전자계산학과 졸업 학사
2002년 3월~현재 : 단국대학교 전자계산학과 석사과정
<관심분야> 정보보호, XML 보안

유 선 영 (Sun-Young Yoo) 준회원
2002년 2월 : 단국대학교 수학과 졸업 학사
2002년 3월~현재 : 단국대학교 전자계산학과 석사과정
<관심분야> 정보보호

박 창 섭 (Chang-Seop Park) 정회원
1983년 : 연세대학교 경제학과 졸업
1983년 : 한국 IBM 근무
1990년 : 미국 Lehigh Univ. 전자계산학 박사
1990년~현재 : 단국대학교 전자컴퓨터학부 교수
<관심분야> 부호이론, 암호학