정보 분산 알고리즘을 이용한 PCC 기법의 개선
(An Improvement of PCC Scheme by using Information Dispersal Algorithm)

현 상 원 † 박 용 수 † 조 유 근 ††
(Sangweon Hyun) (Yongsu Park) (Yookun Cho)

요 약 본 논문에서는 최근에 발표된 스트림 인증 기법중 하나인 PCC 기법 [1]을 정보 분산 알고리즘을 이용하여 개선한 방법을 제시한다. PCC 기법에서는 수신된 스트림 데이타의 검증이 패킷의 그룹 단위로 수행되는데 이때 관련된 서명 패킷이 수신되어야만 그룹에 속한 패킷들이 검증될 수 있다는 약점을 갖는다. 본 논문에서는 PCC 기법의 서명 패킷에 정보 분산 알고리즘을 적용하여 생성되는 단편들을 그룹에 속한 전체 패킷들에 포함시켜 전송하며, 수신자가 이중 정해진 수 이상의 패킷을 수신했을 경우 서명 패킷의 복원이 가능하도록 함으로써 원기법의 문제점을 해결했다. 뿐만 아니라 시뮬레이션 결과의 분석에 의하면 제안 기법은 관련 최신 기법인 SAIDA [2]보다 검증 확률이 높고, 구현 프로그램의 수행 시간을 측정한 결과 계산 시간 면에서 더 나은 성능을 보였다.

키워드 : 보안, 암호, 전자 서명, 스트림 배포

Abstract We propose an efficient stream authentication scheme that is an improvement of PCC scheme by using information dispersal algorithm. The drawback of PCC scheme is that received packets for each group are verifiable only if the signature packet of the group is successfully received. The proposed scheme processes the signature packet by introducing some amount of redundancy and splitting the result into pieces, which are then transmitted. The receiver is able to reconstruct the signature packet if the number of the received pieces is larger than the threshold. It is shown that under the same communication overhead verification probability of the proposed scheme is higher than that of SAIDA. Moreover, its computational cost is lower than that of SAIDA.

Key words : security, cryptography, digital signature, stream distribution

1. 서 론

인터넷 사용자가 늘어나고 망 대역폭이 커짐에 따라 인터넷을 통하여 오디오나 비디오 같은 스트림 데이타를 제공하는 서비스가 늘어나고 있다. 또한, 인터넷이 상업적인 용도로 이용되면서 이런 서비스들도 점차 유료화되고 있는 추세이다.

상업적인 스트림 서비스가 널리 사용되기 위해서는 보안이 무엇보다도 중요하다. 일례로, 수신자는 뉴스나 교통 정보 스트림이 올바른 방송국으로부터 온 것임을, 그리고 임의의 해커에 의해 이들이 변조되지 않았음을 확인하고 싶어 할 것이다. 이렇듯, 여러 가지 보안 요소 중 출처 인증(source authentication)과 데이타 무결성(data integrity)을 제공하는 인증 기법이 무엇보다 중요하다.

실시간으로 생성/전달되는 스트림 데이타를 인증하는 문제는 일반 데이타를 인증하는 것과 다르며, 다음과 같은 문제점을 해결해야한다. 첫째, 서명 연산량을 최소화해야 한다. 전자 서명이 많은 계산량을 요구하는 작업임을 생각해볼 때, 모든 데이타를 서명하는 것은 송신 서버에게 상당한 부하를 준다. 둘째, 통신 오버헤드의 크기를 최소화해야 한다. 셋째, 스트림 데이타는 다양한 환경에서 다양한 경로를 통해 전송되므로 경우에 따라서 많은 손실이 발생한다[3]. 따라서, 인증 기법은 데이타 손실에도 충분히 높은 검증 확률(수신된 패킷들 중 검증 가능한 패킷 수의 비[2])을 제공해야 한다.

스트림 데이타의 인증에 관한 연구에는 스트림 데이타에 적합한 매우 효율적인 서명 기법을 연구한 내용과, 데이타를 그룹으로 묶은 후 서명 연산을 적용하여 많은

† 비 회 원 : 서울대학교 전기.컴퓨터 공학부
swhyun@ssrnet.snu.ac.kr
yspark@ssrnet.snu.ac.kr
†† 종신회원 : 서울대학교 컴퓨터공학부 교수
cho@ssrnet.snu.ac.kr
논문접수 : 2003년 11월 24일
심사완료 : 2004년 11월 19일

계산량을 요구하는 서명 연산의 횟수를 줄인 내용이 있으며(2 절 참조), 본 논문에서는 후자를 다룬다.

본 논문에서 제안하는 기법은 PCC 기법[1]에 기반하며 Rabin의 정보 분산 알고리즘(Information Dispersal Algorithm)[4]을 이용해서 PCC 기법의 약점을 개선했다. PCC 기법에서는 그룹에 포함된 패킷들의 인증을 위해 서명 패킷을 보내며 이것이 손실될 경우 수신자는 해당 그룹 내 모든 패킷들을 검증할 수 없다. 이에 제안 기법에서는 서명 패킷에 정보 분산 알고리즘을 적용하고 그때 생성되는 단편들을 그룹에 속한 전체 패킷들로 나누어서 전송한다. 수신자는 그룹 내 패킷들 중 정해진 수 이상을 수신한 경우 서명 패킷의 복원이 가능하도록 설계했다.

제안된 기법의 장점을 열거하면 다음과 같다. 첫째, PCC 기법이나 GM의 기법[5], EMSS[3], Piggybacking 기법[6]과는 달리 서명 패킷을 가지지 않는다. 둘째, 패킷 생성을 시뮬레이션하여 실험한 결과 PCC 기법과 관련 최신 기법인 SAIDA 기법[2]보다 높은 검증 확률을 보였다. 일례로, 패킷 당 인증 부가 정보량이 34바이트일 때, 제안 기법의 검증 확률은 실험된 모든 패킷 손실율에 대해서 92%이상이었으나, PCC 기법의 경우 패킷 손실율이 높아짐에 따라 75%까지, SAIDA 기법의 경우 60%까지 검증 확률이 떨어졌다. 셋째, 송신 서버 및 수신자의 수행 속도가 SAIDA 기법보다 빨랐으며, 송신 서버의 경우 약 28%, 수신자의 경우 약 10% 더 빨랐다.

본 논문의 구성은 다음과 같다. 먼저 2절에서 관련 연구를 서술한 후, 3절에서 PCC 기법과 개선 기법을 제시한다. 4절에서는 제안 기법의 검증 확률에 대한 수학적 분석 결과와 실험을 통하여 각 기법의 검증 확률을 분석한 결과를 설명한다. 또 제안 기법과 SAIDA 기법의 송신 서버 및 수신자 계산량을 비교한다. 마지막으로, 5절에서 결론을 맺는다.

2. 관련 연구

스트림 데이타의 인증에 관한 연구는 크게 2가지로 나뉜다. 첫째, 스트림 데이타에 적합한 매우 효율적인 서명 기법을 연구한 내용과, 둘째, 데이타를 그룹으로 묶은 후 서명 연산을 적용하여 많은 계산량을 요구하는 서명 연산의 횟수를 줄인 내용이다. 이들은 서로 독립적이며 병행해서 사용하면 보다 좋은 결과를 얻을 수 있다. 본 절에서는 후자의 접근 방식을 채택한 방법들을 살펴본다. 전자의 접근 방식에 관한 연구는 [5]에 설명되어 있다.

데이타를 그룹으로 묶는 방법으로는 Gennaro와 Rohatgi가 제안한 방법으로 오프라인 방식이 있다[7]. 이 기법에서 패킷 그룹 내 k번째 패킷은 k+1번째 패킷을 해쉬 함수에 입력한 결과값을 가지고 있으며 그룹 내 첫 번째 패킷만이 서명되어 전체적으로 서명의 횟수가 상당히 줄어든다(해쉬 체인 기법[1]을 사용). 하지만, 이 기법은 패킷 데이타가 한 개라도 전송 도중 유실될 경우 그룹 내 그 이후의 패킷에 대해 검증할 수 없으며 그룹의 마지막 패킷부터 처음 패킷까지 역방향으로 계산하여야 한다. 따라서, 첫 패킷부터 차례대로 데이타를 생성하여 전송하는 온라인 스트림 데이타의 성질과 잘 맞지 않으며 오프라인 방식에 적합하다. 또한 수신자측의 성공적인 검증을 위해서 서명 패킷의 손실은 없다는 가정이 필요하다.

Wong과 Lam은 Merkle이 고안한 인증 트리 기법[8]을 이용하여 데이타를 그룹으로 묶어 서명하였다[9]. 이 기법에서 송신 서버는 먼저 해쉬 함수를 이용하여 트리 구조를 만든 후, 루트 노드를 서명한다. 각 패킷은 서명값과 트리 내 자신에게 해당하는 잎 노드부터 루트 노드까지의 경로 중 형제값을 모두 가진다. 이 기법의 장점은 그룹 내 패킷을 단 1개만 수신하여도 수신자는 서명을 확인할 수 있다. 하지만, 각 패킷에 포함되는 인증 부가 정보량이 상당히 크다. 또한, 그룹의 크기가 송신 서버의 패킷 버퍼 크기로 제한되어 그룹의 크기를 늘리는 데 제약점을 가지며, 그룹 내 모든 패킷의 인증 정보가 한꺼번에 계산되어서 송신 서버가 보내는 데이타 패턴이 간헐적(bursty)으로 배출되는 단점이 있다.

Perrig, Canetti, Song, 그리고, Tygar는 TESLA (Timed Efficient Stream Loss-tolerant Authentication)와 EMSS라는 두 가지 기법을 제안하였다[3]. TESLA는 MAC(Message Authentication Code)을 이용하여 데이타를 인증하며, MAC을 생성하는 데 사용되는 키 값은 일정 시간 후에 공개된다. 이 기법은 송신 서버의 계산량이 매우 적고, 인증 부가 정보량이 적은 등 상당히 효율적이나 송신 서버와 수신자 사이에 클럭 값이 일정 오차 내로 동기화되어야 하며, 데이타의 전송 지연 시간의 한계값을 모든 수신자가 알고 있어야 하는 등 여러 가지 제약 조건을 가지고 있어 응용 범위가 제한된다. 또한, 이 기법은 부인 방지(*non-repudiation*)를 제공하지 못한다.

EMSS는 k 번째 패킷의 해쉬값이 k+1 번째 이후의 여러 패킷에 포함되며, 서명 패킷은 그룹 내 마지막 패킷들의 해쉬값들과 이를 서명한 서명값을 가지고 있다(해쉬 체인 기법을 사용). 이 기법은 매우 간단하며 데이타 유실이 발생하여도 검증 확률이 상당히 높으나, k

1) 본 논문에서 해쉬 체인 기법은 한 패킷의 해쉬값을 다음 패킷에 저장하고, 저장된 패킷의 해쉬값을 그 다음 패킷에 저장하는 방식을 말한다.

번째 패킷의 해쉬값을 가지는 패킷들이 난수로 결정된다. 따라서 이 기법은 결정적(deterministic)이지 못하다. 또 서명 패킷 손실 시 수신된 모든 패킷의 검증이 불가능하다. 같은 논문에서 저자는 정보 분산 알고리즘[4]을 사용한 확장된 EMSS를 제안하였다. 이 기법은 검증 확률이 경우에 따라서 EMSS보다 높지만 역시 위에서 언급한 문제를 그대로 가지고 있다.

Golle와 Modadugu는 일반적으로 인터넷에서 패킷 손실이 연속적으로 일어난다는 사실에 주목하였으며, 이를 고려한 GM의 기법을 제시하였다[5]. 이 기법은 해쉬 체인 기법을 기반으로 하며, 송신 서버의 패킷 버퍼 메모리와 해쉬 버퍼 메모리가 제한되어있다고 가정할 때, 최소량의 인증 정보로 최대한의 연속된 패킷 손실을 견딜 수 있게끔 설계하였다. 또한, 이 기법은 결정적인 방법이며, 송신 서버의 계산량이 매우 작다. 하지만, 이 기법은 인증 정보량을 조절할 수가 없어서 검증 확률을 원하는 만큼 얻을 수 없으며, 또 서명 패킷 손실시 수신된 모든 패킷의 검증이 불가능하다.

Miner와 Staddon은 여러 번의 연속된 패킷 손실 발생시에도 높은 검증 확률을 제공할 수 있는 Piggybacking 기법을 제안하였다[6]. 이 기법도 해쉬 체인 기법을 기반으로 하며, 인증 단위가 되는 그룹에 속한 패킷들을 같은 크기를 갖는 여러 개의 우선순위 클래스들로 나누어 처리함으로서 각 클래스에 속한 패킷들에 대해서 검증 확률에 차등을 두는 것이 가능하도록 하였다. 따라서 디지털 스트림을 인코딩할 때 나타나는 정보들의 우선순위 특성을 검증 확률에 반영할 수 있다. 또한 이 기법도 결정적인 방법이다. 하지만 다른 패킷들에 비해 상대적으로 서명 패킷의 크기가 상당히 커지며, 또 서명 패킷 손실시 수신된 모든 패킷의 검증이 불가능하다는 문제점을 갖는다.

Park, Chong 그리고 Siegel은 패킷 손실 발생시 인증 과정에서 나타나는 문제점을 해결하기 위해 정보 분산 알고리즘을 이용하는 SAIDA(Signature Amortization using IDA) 기법을 제안하였다[2]. SAIDA 기법은 그룹별로 인증이 이루어지며 이를 위해 그룹에 포함된 전체 패킷들의 해쉬값과 그들을 연결한 값의 서명값을 계산한다. 정보 분산 알고리즘은 해쉬값들을 연결한 값과 서명값에 적용되며 결과 단편들이 전송되는 패킷에 하나씩 포함된다. 수신자측은 전송된 전체 패킷들중 정해진 개수 이상을 수신했을 경우 이 정보들을 성공적으로 복원할 수 있으며, 이를 이용해서 인증 과정을 수행한다. 따라서 별도의 서명 패킷 없이 손실 환경에서 인증을 위한 정보의 효과적인 전송이 가능하다. 또 요구되는 검증 확률에 따라 패킷당 오버헤드를 조절하는 것이 용이하며 기존의 다른 기법들과 비교했을 때 우수한 검증 확률을 제공한다. SAIDA 기법의 단점은 패킷을 전송하는 시점이 그룹에 포함된 전체 패킷들에 대한 해쉬값을 계산하고, 정보 분산 알고리즘을 적용한 이후가 되기 때문에 송신 서버측의 지연 시간이 길고 필요한 패킷 버퍼의 크기도 크다는 점이다.

3. PCC(Park-Chung-Cho) 기법의 개선

먼저, 3.1 절에서 PCC 기법에 대해 설명하고 이를 기반으로 하여 3.2 절에서 개선된 제안 기법을 설명한다. 본 논문에서 사용하는 용어의 의미는 다음과 같다. h(x)는 일방향 해쉬 함수이다. $Sig_A(M)$은 데이타 M을 서명자 A의 개인키로 전자 서명한 서명값이며, A의 공개키를 이용하여 $Sig_A(M)$을 복호한 값과 h(M)이 일치하면 M이 성공적으로 검증된다. 본 논문에서 서명자는 스트림 데이타를 전송하는 송신 서버이며 SIG(M)은 M에 대한 송신 서버의 서명을 뜻한다. 또한, C||D는 데이타 C와 D를 연결(concatenate)시킨 것, |E|는 데이타 E의 바이트 단위 크기를 그리고, e|f는 f가 e의 배수임을 뜻한다.

3.1 PCC 기법

스트림 데이타 $M_0, M_1, ..., M_{tn-1}$ (n>0, t>0)이 있어서 송신 서버는 처음 n개의 M_i ($0 \le i < n$)에 대해 (이들을 그룹 G_0라 부른다) 인증 정보를 생성한다. G_0 내 각 M_i에 대하여 패킷 $P_i=(M_i$, 인증 정보)를 생성하고 수신자에게 전송한다. 송신 서버는 모든 패킷 P_i를 전송한 후, 서명 패킷을 생성, 전송한다. 송신 서버는 위 작업을 $G_1=\{M_{n+i} | (0 \le i < n)\}$, $G_2=\{M_{2n+i} | (0 \le i < n)\}$, ..., $G_{t-1}=\{M_{(t-1)n+i} | (0 \le i < n)\}$에 대해 차례로 반복 수행한다. 수신자는 특정 그룹에 해당하는 패킷들과 서명 패킷을 받고 검증한다. 각 패킷에는 인증 정보로 해쉬값이 포함되며 그 개수는 l이다. 또한, 서명 패킷은 n_s개의 해쉬값을 연결한 값과 이에 대한 서명값을 가지고 있다. 그리고, 송신 서버는 패킷 버퍼를 가지고 있어서 패킷을 한꺼번에 처리할 수 있으며, 그 크기는 2^b보다 같거나 크다고 가정한다. PCC 기법에서는 $n_s | n$, $2^{l-1} | (n/n_s)$이어야 하며, 이 조건을 만족시키는 n, n_s의 선택에 대한 자세한 내용은 [1]에 나와 있다.

인증 트리 각 그룹마다 n_s개의 인증 트리가 있다. 인증 트리 A_m ($0 \le m < n_s$)의 잎 노드 수는 n/n_s이다. A_m의 루트 노드는 $n/(2^{l-1}n_s)$개의 완전 이진 서브 트리 T_k ($nm/(2^{l-1}n_s) \le k < n(m+1)/(2^{l-1}n_s)$)의 루트 노드를 자식으로 가지며, 각 서브 트리 T_k는 잎 노드의 수가 2^{l-1}개이다(그림 1 참조). A_m의 각 노드 e에는 하나의 값 E가 연관되어 있으며, 각 M_i마다 하나의 잎 노드 $e_{c\sim c}$가 대응된다. 또 이에 연관된 값은 $E_{c\sim c}=h(M_i)$로 설

정한다. 또 $e_{c\sim c}$는 하나의 서브트리 T_a에 속해있고, T_a는 T_b (b=a+1 혹은 a-1)와 쌍을 이룬다. e가 내부 노드인 경우, $e_{c\sim c'}$ (c<c')로 표시하며 자식 노드 $e_{c1\sim c1'}$, $e_{c2\sim c2'}$, ..., $e_{cj\sim cj'}$, ($c_1 \le c_1' < c_2 \le c_2' < ... < c_j \le c_j'$)를 가질 때 $c=c_1$, $c'=c_j'$로 설정된다. 또한, $e_{c\sim c'}$에 연관된 값은 $E_{c\sim c'}=h(E_{c1\sim c1'}||E_{c2\sim c2'}||...||E_{cj\sim cj'})$이며, 일례로, 그림 1에서 $E_{0\sim1}=h(E_{0\sim0}||E_{1\sim1})$이다. 우리는 노드 e에서 조상 노드 e'까지 경로 상에 있는 각 노드의 모든 형제 노드와 연관된 값의 집합을 Siblings(e, e')라고 부르며, 일례로, 그림 1에서 Siblings($e_{0\sim0}$, $e_{0\sim1}$)={$E_{1\sim1}$}이다. 서브 트리의 쌍 $Pair_j=(T_{2j}, T_{2j+1})$ ($nm/(2^l n_s) \le j < n(m+1)/(2^l n_s)$)을 정의하며, 일례로 그림 1에서 $Pair_0=(T_0,T_1)$, $Pair_1=(T_2,T_3)$이다. 또한, 루트 노드와 연관된 값은 모든 서브 트리 T_k의 루트 노드와 연관된 값을 연결한 값의 해쉬값이다. 또한, 패킷 손실이 연속적으로 발생하는 특성을 띠는 인터넷 환경에서 높은 검증 확률을 제공하기 위해 인증 트리의 i번째 잎 노드는 $f(i)$ ($=(i \bmod 2^l)(2^{b-l}) + \lfloor i/2^l \rfloor$ 번째 데이터와 연관된다.

예 1. 그림 1은 l=2, n=32, n_s=4 그리고, 2^b=8 일 때, G_0에 대한 인증 트리 4개 중 하나인 A_0의 구조를 보여주고 있다. 인증 트리의 루트 노드 $e_{0\sim7}$은 잎 노드가 2^{l-1}개인 완전 이진 서브 트리 T_0, T_1, T_2, T_3의 루트 노드를 자식으로 가지며, 루트 노드와 연관된 값 $E_{0\sim7}$은 $h(E_{0\sim1}||E_{2\sim3}||E_{4\sim5}||E_{6\sim7})$이다.

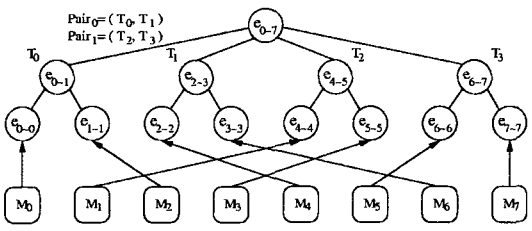

그림 1 PCC 기법에서 사용하는 인증 트리의 예

패킷 생성 과정 송신 서버는 처음 n개의 스트림 데이타 M_i ($0 \le i < n$)에 대해 (즉 G_0에 대해), n_s개의 인증 트리 A_m ($0 \le m < n_s$)을 다음과 같이 생성한다: 먼저, n/n_s개의 스트림 데이타를 이용하여 인증 트리 A_m의 모든 잎 노드와 이에 연관된 값을 계산한 후, A_m의 모든 완전 이진 서브 트리의 내부 노드와 이에 연관된 값을 계산한 다음, A_m의 루트 노드와 이에 연관된 값을 계산한다. 송신 서버는 각 패킷 $P_i=(M_i$, Siblings($e_{c\sim c}$, T_a의 루트 노드)$\cup${T_b의 루트 노드와 연관된 값})를 생성하며 수신자에게 전송한다. 모든 A_m ($0 \le m < n_s$)을 만들고 이에 대응하는 패킷들을 전송한 후, 서명 패킷을 생성, 전송한다. 단, 서명 패킷의 내용은 모든 A_m ($0 \le m < n_s$)의 루트 노드에 연관된 값을 연결한 값과 이를 서명한 값이다. 송신 서버는 위 작업을 G_1, G_2, ..., G_{t-1}에 대해 반복 수행한다.

예 2. l=2, n=32, n_s=4 그리고, 2^b=8 일 때, 송신 서버는 G_0에 대하여 예 1의 인증 트리 A_0를 포함한 4개의 인증 트리를 생성한 후, 32개의 패킷을 생성, 전송한다. 일례로, M_2는 잎 노드 $e_{1\sim1}$에 대응되며, $e_{1\sim1}$은 T_0에 속하고 T_0는 T_1과 쌍을 이룬다. 따라서, 패킷 P_2는 (M_2, {$E_{0\sim0}$, $E_{2\sim3}$})이다. 그 후, 서명 패킷 ($E_{0\sim7}||E_{8\sim15}||E_{16\sim23}||E_{24\sim31}$, $SIG(E_{0\sim7}||E_{8\sim15}||E_{16\sim23}||E_{24\sim31})$)을 전송한다.

패킷 검증 과정 서명 패킷 손실 시에는 수신된 모든 패킷의 검증이 불가능하며, 서명 패킷을 성공적으로 수신했을 경우의 검증 과정은 다음과 같다. 수신자가 특정 그룹 G_r ($0 \le r < t$)에 속하는 일부 패킷들을 수신하였을 때, 검증 과정은 인증 트리 A_m 단위로 이루어지며, 수신자는 각 A_m에 연관된 패킷들을 모두 검증하거나 검증하지 못한다. 수신자가 패킷 P_i를 수신하면, 패킷 내 인증 정보를 이용하여 M_i에 대응하는 잎 노드 $e_{c\sim c}$를 포함하는 서브 트리 T_a의 루트 노드와 연관된 값과, T_a와 같은 서브 트리 쌍에 속하는 트리 T_b의 루트 노드와 연관된 값을 구할 수 있다. 수신자가 A_m에 대한 각 서브 트리 쌍 $Pair_j$ ($nm/(2^l n_s) \le j < n(m+1)/(2^l n_s)$)의 잎 노드에 대응하는 2^l개의 패킷 중 적어도 1개의 패킷을 수신했으면, A_m의 모든 서브 트리의 루트 노드와 연관된 값을 계산할 수 있으며, 따라서 A_m의 루트 노드에 연관된 값을 계산할 수 있다. 이 값이 서명 패킷 내 송신 서버가 만든 값과 일치하면 성공적으로 검증된 것이며, 일방향 함수의 성질에 따라 이 값을 계산할 때 사용한 모든 T_k의 루트 노드와 관련된 값이 검증된 것이고, A_m의 잎 노드에 대응되는 각 패킷들도 성공적으로 검증된 것이다. 하지만, 특정 서브 트리 쌍에 대응하는 2^l개의 패킷이 모두 손실된 경우 A_m의 루트 노드에 연관된 값을 구할 수 없다. 이러한 경우의 발생을 줄이기 위해 식 $f(i)=(i \bmod 2^l)(2^{b-l})+\lfloor i/2^l \rfloor$에 의해 연속된 스트림 데이타를 서브 트리 쌍마다 하나씩 대응하게 했다.

예 3. 예 2에서 수신자가 P_0, P_2, P_4, P_6 중 P_2만을 수신했다고 하자. P_2의 인증 정보는 {$E_{0\sim0}$, $E_{2\sim3}$}이며, $E'_{1\sim1}=h(M_2)$을 계산하여 $E'_{0\sim1}=h(E_{0\sim0}||E'_{1\sim1})$을 계산할 수 있다. 이 때, P_1, P_3, P_5, P_7이 모두 손실된 경우, $E_{4\sim5}$, $E_{6\sim7}$을 구할 수 없어서 P_2를 검증할 수 없다. 만일 P_5를 추가로 수신한 경우 이 패킷의 인증 정보는 {$E_{7\sim7}$, $E_{4\sim5}$}이며, $E'_{6\sim6}=h(M_5)$을 계산하여 $E'_{6\sim7}=h(E'_{6\sim6}||E_{7\sim7})$을 계산할 수 있다. 최종적으로 $E'_{0\sim7}=h(E'_{0\sim1}||E_{2\sim3}||E_{4\sim5}||E'_{6\sim7})$을 계산할 수 있으며, $E'_{0\sim7}$와 수신된 서명 패킷에 포함된 $E_{0\sim7}$을 비교함으로서 P_2, P_5를 모두 검증할

수 있다.

3.2.제안 기법

PCC 기법은 서명 패킷이 손실될 경우 수신된 모든 패킷의 검증이 불가능하다는 문제점을 갖는다. 이에 [1,3]에서는 패킷 손실이 연속적으로 발생하는 인터넷의 전송 특성을 고려하여 서명 패킷을 시간차를 두어 반복해서 전송하는 방법을 언급했다. 하지만 이 방법은 동일한 서명 패킷의 반복 전송으로 인해 통신 오버헤드가 증가되며, 검증 지연 시간이 증가한다는 문제점을 갖는다. 이러한 문제점을 해결하면서 서명 패킷의 내용을 효과적으로 전송하기 위해 본 제안 기법에서는 Rabin의 정보 분산 알고리즘 [4]을 사용한다.

정보 분산 알고리즘은 정보 분산 연산(*Dispersal*())과 정보 복원 연산(*Recovery*()) 두 가지로 구성된다. 데이타 F에 대해 *Dispersal*(F, m', n)을 적용하면 n개의 단편 F_i $(0 \le i < n)$들이 생성되며, 이중 임의의 m'($\le$n)개의 단편에 대해 *Recovery*($\{F_j | (0 \le j < m'), (0 \le i < n)\}, m', n$)을 적용하면 F가 복원된다. 즉, n-m'개 단편의 손실에 대해서는 저항력을 갖는다. 각 단편의 크기는 $|F|/m'$이다.

패킷 생성 과정 송신 서버는 처음 n개의 스트림 데이타 M_i $(0 \le i < n)$에 대해 (즉 G_0에 대해), n_s개의 인증 트리 A_m $(0 \le m < n_s)$을 PCC 기법에 의해 생성한다. 모든 A_m $(0 \le m < n_s)$이 만들어지고 나면 서명 패킷 SIG=($E_{0\sim n/n_s-1} || E_{n/n_s\sim 2n/n_s-1} || \ldots || E_{(n_s-1)n/n_s\sim n-1}$), SIG=($E_{0\sim n/n_s-1} || E_{n/n_s\sim 2n/n_s-1} || \ldots || E_{(n_s-1)n/n_s\sim n-1}$))를 생성한다. 생성된 서명 패킷에 대해서 정보 분산 연산 *Dispersal*(SIG, m', n)을 적용하며, 그 결과는 서명 패킷에 대한 n개의 단편들 SIG_i $(0 \le i < n)$가 된다. 송신 서버는 각 패킷 P_i=(M_i, Siblings($e_{c\sim c}$, T_a의 루트 노드)$\cup${T_b의 루트 노드와 연관된 값}, SIG_i)를 생성하며 수신자에게 전송한다. 따라서 송신 서버에 필요한 패킷 버퍼의 크기 2^b=n이다. 송신 서버는 위 작업을 G_1, G_2, ..., G_{t-1}에 대해 반복 수행한다.

예 4. l=2, n=32, n_s=4, m'=16 그리고 2^b=n일 때의 패킷 생성 과정은 다음과 같다. 해쉬 함수와 전자 서명 알고리즘으로 [2]에서 사용한 128비트 MD5, 1024비트 RSA를 선택하면, |h()|=16, |SIG()|=128이다. 송신 서버는 G_0에 대하여 예 1의 인증 트리 A_0를 포함한 4개의 인증 트리를 생성한 후, 32개의 패킷 P_i $(0 \le i < 32)$를 생성한다. 그 후, 서명 패킷 SIG=($E_{0\sim7} || E_{8\sim15} || E_{16\sim23} || E_{24\sim31}$, SIG($E_{0\sim7} || E_{8\sim15} || E_{16\sim23} || E_{24\sim31}$)) (|SIG|=192바이트)를 생성하고 정보 분산 연산을 적용한다. 그 결과는 32개의 단편들 SIG_i $(0 \le i < 32)$가 되며, 각 단편의 크기는 12바이트(=192/16)가 된다. 그림 2는 인증 트리 A_0에 대응되는 패킷들의 생성 예이다. 일례로, 전송되는 패킷들 중 하나인 P_8은 (M_8, {$E_{0\sim0}$, $E_{2\sim3}$}, SIG_8)가 된다.

패킷 검증 과정 수신자가 특정 그룹 G_r $(0 \le r < t)$에 속하는 전체 n개의 패킷들 중 m'개 미만의 패킷만을 수신할 경우에는 정보 복원 연산을 통한 서명 패킷의 복원이 불가능하므로 검증이 실패한다. 수신자가 n개의 패킷들 중 m'개 이상의 패킷들을 수신했을 경우에는 정보 복원 연산 *Recovery*($\{F_j | (0 \le j < m'), (0 \le i < n)\}, m', n$)을 통해 서명 패킷이 복원되며, 이후의 검증 과정은 PCC 기법과 동일하다. 수신자가 인증 트리 A_m에 대한 각 서브 트리 쌍 $Pair_j$ $(nm/(2^l n_s) \le j < n(m+1)/(2^l n_s))$의 잎 노드에 대응하는 2^l개의 패킷 중 적어도 1개의 패킷을 수신했으면, A_m의 모든 서브 트리의 루트 노드와 연관된 값을 계산할 수 있으며, 따라서 A_m의 루트 노드에 연관된 값을 계산할 수 있다. 이 값이 복원된 서명 패킷 내 송신 서버가 만든 값과 일치하면 성공적으

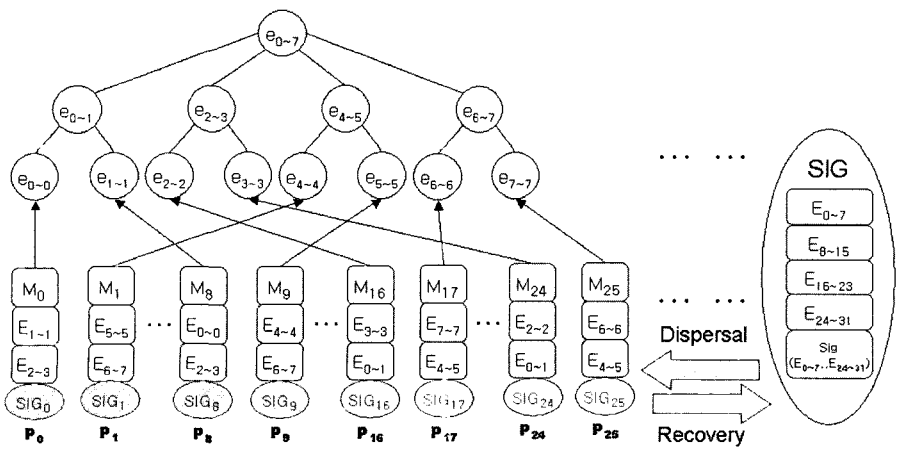

그림 2 제안 기법에서 패킷 생성 및 검증 예

로 검증된 것이며, 일방향 함수의 성질에 따라 이 값을 계산할 때 사용한 모든 T_k의 루트 노드와 관련된 값이 검증된 것이고, A_m의 잎 노드에 대응되는 각 패킷들도 성공적으로 검증된 것이다. 하지만, 특정 서브 트리 쌍에 대응하는 2'개의 패킷이 모두 손실된 경우 A_m의 루트 노드에 연관된 값을 구할 수 없으므로 검증이 실패한다.

예 5. 예 4에서 수신자가 G_0에 속한 32개의 패킷들 중 16개 이상의 패킷들을 수신했으며, 수신된 패킷에 P_0, P_8, P_{16}, P_{24} 중 P_8만 포함되어 있다고 가정하자. 수신자는 우선 16개의 정보 분산 연산 단편에 $Recovery(\{SIG_i\}(0\le j<16),(0\le i<32)\},16,32)$을 적용해서 서명 패킷을 복원한다. P_8 패킷의 인증 부가 정보는 $\{E_{0\sim0}, E_{2\sim3}\}$이며, $E'_{1\sim1}=h(M_8)$을 계산하여 $E'_{0\sim1}=h(E_{0\sim0}||E'_{1\sim1})$을 계산할 수 있다. 이 때, P_1, P_9, P_{17}, P_{25}가 모두 손실된 경우, $E_{4\sim5}$, $E_{6\sim7}$을 구할 수 없어서 P_8을 검증할 수 없다. 하지만 이 네 패킷 중 적어도 하나 이상을 수신한 경우 성공적으로 검증할 수 있다. 예를 들어, P_{17}을 수신한 경우 이 패킷의 인증 부가 정보는 $\{E_{7\sim7}, E_{4\sim5}\}$이며, $E'_{6\sim6}=h(M_{17})$을 계산하여 $E'_{6\sim7}=h(E'_{6\sim6}||E_{7\sim7})$을 계산할 수 있다. 최종적으로 $E'_{0\sim7}=h(E'_{0\sim1}||E_{2\sim3}||E_{4\sim5}||E'_{6\sim7})$을 계산할 수 있으며, $E'_{0\sim7}$와 복원된 서명 패킷에 포함된 $E_{0\sim7}$을 비교함으로서 수신된 패킷 P_8, P_{17}을 모두 검증할 수 있다.

4. 성능 분석

먼저, 4.1절에서는 제안 기법의 검증 확률을 기존 기법들과 비교 실험한 결과를 설명하며, 4.2절에서는 제안된 기법의 검증 확률을 해석적으로 분석한다. 4.3절에서는 제안 기법과 기존 기법들에 대하여 송신 서버 및 수신자의 계산량을 비교한다. 해쉬 함수와 전자 서명은 [2]에서 사용한 알고리즘인 128비트 MD5, 1024비트 RSA를 선택했으며, 비교 대상 기법은 PCC 기법과 SAIDA 기법이다.

4.1 시뮬레이션 결과 및 분석

본 절에서는 시뮬레이션을 통해 각 기법들의 검증 확률을 분석한 결과를 설명한다. 시뮬레이션 환경은 다음과 같다. PCC 기법의 경우 두개의 서명 패킷이 128개의 패킷만큼 지연 후 반복 전송되는 것으로 가정했다. 손실이 연속적으로 발생하는 인터넷의 전송 특성을 시뮬레이션하기 위해 패킷 손실은 2-상태 마르코프 모델에 따라 발생시켰다[1-3]. [2]에서와 마찬가지로 다음의 시뮬레이션 인자들을 사용했다: 그룹의 크기 n=128, 패킷 손실율 20%, 연속적으로 손실된 패킷 수의 평균값을 8로 설정했다. 실험 결과는 시뮬레이션을 10^5 번 수행하여 나온 결과들의 평균값이다.

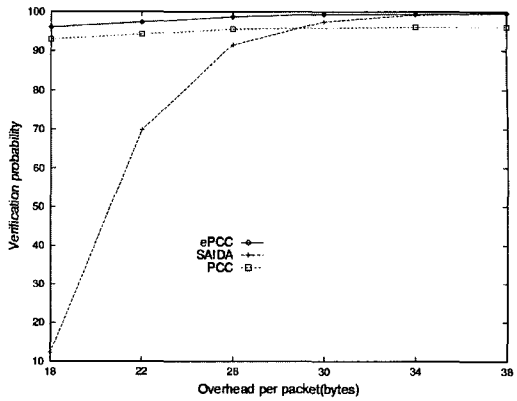

그림 3 패킷 당 인증 정보량의 변화에 따른 검증확률 결과

그림 3은 패킷 당 인증 정보량을 18바이트부터 38바이트까지 증가시켰을 때 각 기법의 검증 확률을 보여준다. 가로축은 패킷 당 인증 정보의 크기이며 세로축은 검증 확률이다. 실험 결과를 보면 제안 기법(ePCC)이 더 작은 인증 정보량으로도 더 높은 검증 확률을 제공함을 확인할 수 있다. PCC 기법의 검증 확률은 약 29바이트까지는 SAIDA 기법보다 높았으나 그 이후부터는 더 낮은 검증 확률을 보였는데, 이는 서명 패킷 손실의 영향이 두드러지기 때문으로 추정된다. SAIDA 기법도 제안 기법(ePCC)보다 낮은 검증 확률을 보였으며, 이는 모든 경우에 대해서 더 큰 m' 값이 요구되기 때문으로 추정된다.

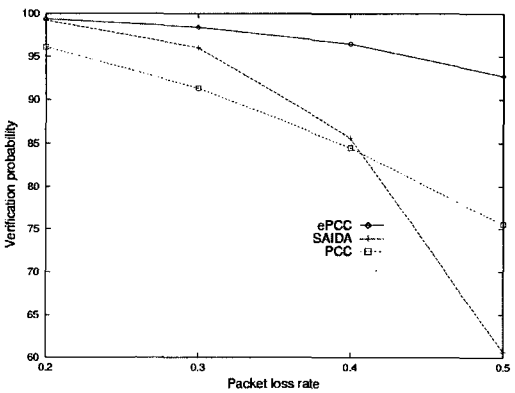

그림 4 패킷 손실율 증가에 따른 검증확률 결과

그림 4는 패킷 당 인증 정보량이 34바이트인 상태에서 패킷 손실율을 20%부터 50%까지 증가시켰을 때 각 기법의 검증 확률을 보여준다. 가로축은 패킷 손실율이며 세로축은 검증 확률이다. 이 경우에도 역시 제안 기법(ePCC)이 가장 좋은 성능을 나타냈으며, 패킷 손실율

이 높아질수록 성능 차이가 점점 커짐을 확인할 수 있었다. 특히 제안 기법(ePCC)은 50%의 패킷 손실율에서도 약 93%의 높은 검증 확률을 보였다.

4.2 검증 확률에 대한 해석적 분석

본 절에서는 패킷 손실이 독립적으로 발생한다고 가정할 때 제안 기법의 검증 확률을 수학적으로 분석한다. 먼저, 패킷 손실율을 $(1-p)$라고 가정하자. 그리고 그룹의 서명 패킷 복원을 위해 필요한 최소 단편의 수는 m'임을 기억하자.

정리 1. 위와 같은 패킷 손실 모델에서 제안 기법의 검증 확률 P_{ePCC}^{ind}는

$$P_{ePCC}^{ind} \geq (1-(1-p)^{2^l})^{n/(2^l n_s)-1} \left\{1-\sum_{i=0}^{m'-n/(2^l n_s)-1}\binom{n-n/n_s}{i}p^i(1-p)^{n-n/n_s-i}\right\}$$

이다.

증명. 제안 기법에서 수신된 패킷 P_k의 성공적인 검증을 위해서는 첫째 P_k에 대응되는 잎 노드가 포함된 인증 트리 A_k의 모든 서브 트리 쌍에 대해서 적어도 하나의 패킷을 수신해야 하며, 둘째 P_k가 속한 그룹 G_j의 서명 패킷을 복원할 수 있어야 한다. 표본 공간 S를 $\{a_0a_1\ldots a_{n-1}|a_i=0,1\,(0\leq i<n)\}$라 하고 S의 각 원소는 G_j에 대응되는 n개 패킷들의 수신 여부를 나타낸다고 하자. 첫째 조건에 해당하는 이벤트를 E_1, 둘째 조건에 해당하는 이벤트를 E_2라고 하면, $E_1, E_2 \subset S$이다. E_1이 발생할 확률은 PCC 기법에서 서명 패킷을 수신했다고 가정했을 때의 검증 확률과 같으므로, [1]의 정리 2에 의해 $P(E_1)=(1-(1-p)^{2^l})^{n/(2^l n_s)-1}$이다. E_1이 발생하면 A_k에 대응되는 n/n_s개의 패킷 중 매 2^l개의 패킷 당 적어도 1개의 패킷은 수신했으므로(3.2 절 참조), 수신된 패킷의 수는 적어도 $n/(2^l n_s)$개 이상이다. 여기서 A_k에 대응되지 않는 $n-n/n_s$개의 패킷 중에서 적어도 $m'-n/(2^l n_s)$개 이상의 패킷들을 수신하는 이벤트를 E_2'라고 하면, $E_1 \cap E_2'$에 속한 경우는 첫째 조건을 만족함은 물론 G_j에 대응되는 n개의 패킷 중 m'개 이상을 수신한 것이 되므로 둘째 조건도 만족하여 검증에 성공한다. $(E_1 \cap E_2') \subseteq (E_1 \cap E_2)$이고, E_1과 E_2'는 서로 독립인 이벤트이므로, $P_{ePCC}^{ind}=P(E_1 \cap E_2) \geq P(E_1 \cap E_2')=P(E_1)P(E_2')$이다.

$P(E_2')$는 [10]의 식 2.5-3에 의해 $1-\sum_{i=0}^{m'-n/(2^l n_s)-1}\binom{n-n/n_s}{i}p^i(1-p)^{n-n/n_s-i}$이므로,

$$P(E_1)P(E_2')=(1-(1-p)^{2^l})^{n/(2^l n_s)-1}\left\{1-\sum_{i=0}^{m'-n/(2^l n_s)-1}\binom{n-n/n_s}{i}p^i(1-p)^{n-n/n_s-i}\right\}$$

이다. □

4.3 송신 서버 및 수신자의 계산량 비교

본 절에서는 PCC 기법, SAIDA 기법 그리고 제안 기법의 송신 서버와 수신자 계산량을 비교 분석한다. 해쉬의 계산량을 C_{hash}, 정보 분산 및 복원 연산의 계산량을 각각 $C_{Dispersal(|F|,m',n)}$, $C_{Recovery(|F|,m',n)}$, 서명 및 서명 확인 연산의 계산량을 각각 C_{sign}, C_{ver}이라고 하자. r은 수신된 패킷수를 나타낸다 ($r \geq m'$). 표 1은 단일 그룹에 대해서 수행되는 패킷 생성/검증의 계산량을 나타낸다 (계산량 산출 방법은 SAIDA는 [2], 제안 기법과 PCC는 [1]를 참조). 표 2는 각 기법을 구현한 후 단일 그룹에 대한 수행 시간을 측정한 결과이다. [2]에 따라 그룹의 크기 n=128로 설정했고 22바이트부터 38바이트까지의 인증 정보량에 대해서 송신 서버와 수신자의 수행 시간을 관찰했다. 실험 환경은 다음과 같다: CPU와 메모리는 각각 Pentium 4 2.4Ghz, 512MBytes이고, 운영 체제, 암호 라이브러리, 컴파일러는 각각 Linux version 2.4.20, Crypto++ 4.2, gcc version 2.96이다. 해쉬 함수와 전자 서명 알고리즘은 128비트 MD5, 1024비트 RSA를 사용했다[2]. 본 실험에서 제안 기법이 PCC 기법보다 더 많은 수행 시간이 걸렸으며, 송신 서버의 경우 약 2배 정도, 수신자의 경우는 약 1.5배 정도 더 많은 시간이 걸렸다. 이는 PCC 기법의 서명 패킷 손실

표 1 송신 서버 및 수신자의 계산량 비교

기 법	그룹에 대한 계산량 (송신 서버)	그룹에 대한 계산량 (수신자)
SAIDA	$(n+1)C_{hash}+C_{Disperse(n\|h()\|+\|SIG()\|,m',n)}+C_{sign}$	$(r+1)C_{hash}+C_{Recovery(n\|h()\|+\|SIG()\|,m',n)}+C_{ver}$
PCC	$(n+(1-1/2^{l-1})n+n_s+1)C_{hash}+C_{sign}$	$(lr+n_s+1)C_{hash}+C_{ver}$
제안 기법	$(n+(1-1/2^{l-1})n+n_s+1)C_{hash}+C_{Disperse(n_i\|h()\|+\|SIG()\|,m',n)}+C_{sign}$	$(lr+n_s+1)C_{hash}+C_{Recovery(n_i\|h()\|+\|SIG()\|,m',n)}+C_{ver}$

표 2 송신 서버 및 수신자의 수행 속도 비교

패킷당 오버헤드 (바이트)	기법	송신 서버 수행 시간(ms)	수신자 수행 시간(ms)
22	SAIDA	62.29	3.64
	PCC	23.33	1.76
	제안 기법	50.32	2.88
26	SAIDA	68.45	3.46
	PCC	23.44	1.78
	제안 기법	53.88	3.04
34	SAIDA	77.68	2.95
	PCC	23.72	1.81
	제안 기법	53.11	2.63
38	SAIDA	80.33	2.85
	PCC	23.93	1.86
	제안 기법	49.48	2.98

문제를 해결하기 위해 제안 기법에서 정보 분산 알고리즘이 수행되기 때문이다. 제안 기법과 SAIDA 기법을 비교해보면, 송신 서버의 경우 제안 기법이 평균 28%정도 수행 속도가 빨랐으며, 수신자의 경우도 제안 기법이 평균 10%정도 더 빨랐다. 이는 해쉬의 계산량은 제안 기법이 많으나 정보 분산 알고리즘의 계산량은 SAIDA가 더 많고, 또 정보 분산 알고리즘에 포함된 유한체 연산의 속도가 해쉬 연산 속도에 비해 훨씬 느리기 때문으로 추정된다.

표 2의 수행 시간 측정 결과에서 제안 기법의 경우 비선형적 특성이 나타남을 확인할 수 있다. 이는 제안 기법의 연산량이 인증 트리와 관계된 인자인 l과 n_s 뿐만 아니라 정보 분산 알고리즘의 인자인 m′ 값에 의해 복합적으로 영향을 받기 때문이다. 제안 기법의 수행 시간에서 중요한 부분을 차지하는 정보 분산 알고리즘의 *Dispersal*() 연산의 수행 시간은 입력의 크기와 m′ 값에 비례하며 (입력의 크기가 더 큰 영향을 미침), *Recovery*() 연산은 $O(m'^2)$의 시간 복잡도를 갖는다[4]. 송신 서버에 대해서 패킷당 오버헤드가 22바이트일 때 제안 기법에서 사용된 인자는 {l, n_s, m′}={1, 16, 64}(검증 확률을 최대로하는 값들의 조합)였다. 표 1의 식을 이용해서 해쉬 연산의 수행 횟수와 *Dispersal*() 연산의 입력의 크기를 계산하면 각각 145번과 384바이트이다. 26바이트일 때 사용된 인자는 {1, 32, 64}였으며, 이 때 해쉬 연산의 수행 횟수와 *Dispersal*() 연산의 입력의 크기는 각각 161번과 640바이트이다. 즉 22바이트일 때보다 26바이트일 때 해쉬 연산의 수행 횟수와 *Dispersal*() 연산의 입력의 크기 모두 더 크며, 그 결과 수행 시간 측정에서도 더 큰 결과를 보였다. 34바이트일 때 사용된 인자는 {1, 32, 36}였다. 이 경우 해쉬 연산의 수행 횟수와 *Dispersal*() 연산의 입력의 크기는 26바이트일 때와 같지만 m′ 값이 36으로 더 작기 때문에 수행 시간도 26바이트일 때보다 더 작았다. 수신자측의 수행 시간은 *Recovery*() 연산의 m′ 값에 비례해서 변화함을 확인할 수 있었다.

5. 결 론

본 논문에서는 최근에 발표된 스트림 인증 기법중 하나인 PCC 기법을 정보 분산 알고리즘을 이용하여 개선한 방법을 제안했다. 제안된 기법은 PCC 기법의 서명 패킷에 정보 분산 알고리즘을 적용함으로서 서명 패킷의 손실 시 수신된 모든 패킷의 검증이 불가능하다는 PCC 기법의 약점을 해결했다. 본 논문에서는 제안된 기법의 검증 확률을 해석적으로 분석한 결과를 제시했다. 기존 기법과 비교해보면 첫째, PCC 기법이나 GM의 기법, EMSS, Piggybacking 기법과는 달리 서명 패킷을 가지지 않는다. 둘째, 시뮬레이션을 통해 검증 확률을 측정한 결과 제안 기법이 기존 기법들보다 매우 높은 검증 확률을 보였다. 일례로, 패킷 당 인증 정보량이 34바이트일 때, 제안 기법의 검증 확률은 전 구간에서 92%이상이었으나, PCC 기법은 패킷 손실율이 높아짐에 따라 75%까지, SAIDA 기법은 60%까지 검증 확률이 낮아졌다. 셋째, 송신 서버 및 수신자의 수행 속도가 SAIDA 기법보다 빨랐으며, 송신 서버의 경우 약 28% 정도, 수신자의 경우 약 10%정도 더 빨랐다.

참 고 문 헌

[1] Yongsu Park, Taesun Chung, and Yookun Cho, An Efficient Stream Authentication Scheme using Tree Chaining, Information Processing Letters, 86(1):1-8, 2003.

[2] Jung Min Park, Edwin K. P. Chong, and Howard Jay Siegel, Efficient Multicast Packet Authentication using Signature Amortization, In IEEE Symposium on Security and Privacy'02, 2002.

[3] Adrian Perrig, Ran Canetti, Dawn Song, and J. D. Tygar, Efficient Authentication and Signing of Multicast Streams over Lossy Channels, In Proceedings of IEEE Security and Privacy Symposium, May, 2000.

[4] Michael O. Rabin, Efficient Dispersal of Information for Security, Load Balancing and Fault Tolerance, Journal of the Association for Computing Machinery, 36(2):335-348, 1989.

[5] Philippe Golle and Nagendra Modadugu, Authenticating Streamed Data in the Presence of Random Packet Loss, In NDSS'01, pages 13-22, 2001.

[6] Sara Miner, Jessica Staddon, Graph-based Authentication of Digital Streams, In IEEE Symposium on Security and Privacy'01, 2001.

[7] Rosario Gennaro and Pankaj Rohatgi, How to Sign Digital Streams, In CRYPTO'97, pages 180-197, 1997.
[8] Ralph C. Merkle, A Certified Digital Signature, In CRYPTO'89, pages 218-238, 1989.
[9] Chung Kei Wong and Simon S. Lam, Digital Signatures for Flows and Multicasts, IEEE/ACM Transactions on Networking, 7(4):502-513, 1999.
[10] Peyton Z. Peebles, Jr., Probability, Random Variables and Random Signal Principles, McGraw-Hill International Edition, 2001.

현 상 원
2002년 2월 성균관대학교 전기전자 및 컴퓨터 공학부 졸업. 2004년 2월 서울대학교 컴퓨터공학부 석사

박 용 수
정보과학회논문지 : 시스템 및 이론
제 31 권 제 2 호 참조

조 유 근
정보과학회논문지 : 시스템 및 이론
제 31 권 제 2 호 참조