인증 트리 기법을 이용한 효율적인 스트림 인증 기법

(An Efficient Stream Authentication Scheme using Tree Authentication)

박 용 수 † 조 유 근 †
(Yongsu Park) (Yookun Cho)

요 약 본 논문에서는 인증 트리 기법을 이용하여 실시간으로 전달되는 스트림 데이타를 인증하는 효율적인 기법을 제시한다. 제시한 기법은 높이가 낮은 인증 트리를 설계하여 송신 서버의 계산량이 매우 작으며, 인증 트리 내 인증 정보를 패킷에 적절히 분배하여 수신자가 데이타를 검증 시 검증 확률이 매우 높다. 본 논문에서는 제안된 기법의 검증 확률을 해석적으로 분석한 결과를 제시하였다. 기존 기법들과 비교해보면 송신 서버의 계산량은 기존 기법 중 가장 우수한 GM의 기법과 거의 비슷한 수준이며, 시뮬레이션을 통해 검증 확률을 측정한 결과 기존 기법들보다 월등히 높은 검증 확률을 보였다.

키워드 : 보안, 인증 프로토콜, 스트림 배포

Abstract We propose an efficient stream authentication scheme using tree authentication. To reduce the computation cost of the sender, we design the authentication tree whose height is very short. We appropriately distribute authentication information over packets so the receiver can verify data with high probability. Moreover, we provide mathematical analysis on the verification probability. For the proposed scheme and previous schemes, we measured the elapsed time for generating authentication information and the proposed scheme has equal to or slightly larger than that of GM's scheme, which has the lowest computation overhead. We performed simulations, which show that the verification probability of the proposed scheme is much higher than that of any other scheme.

Key words : security, authentication protocol, 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, 방송국에서는 임의의 해커가 방송 데이타에 상업적인 내용을 삽입하는 행위를 방지하고 싶어할 것이며, 방송 청취자는 증권 정보나 뉴스가 위/변조되지 않았음을 확인하고 싶어할 것이다. 이렇듯, 여러 가지 보안 요소 중 데이타의 위/변조를 방지하고 부인 방지를 제공하는 인증 기법이 무엇보다 중요하다[7].

실시간으로 생성/전달되는 스트림 데이타를 인증하는 문제는 일반 데이타를 인증하는 것과 다르며, 다음과 같은 문제점을 해결해야한다. 첫째, 데이타의 크기가 상당히 크다. 일반적으로 전자 서명이 많은 계산량을 요구하는 작업임을 생각해볼 때, 모든 데이타를 서명하는 것은 송신자에게 상당한 부하를 준다(이에 비해, 수신자가 데이타를 검증하는 데 필요한 계산량은 매우 작다). 둘째, 수신자는 받은 데이타를 제한된 시간 내에 검증할 수 있어야 한다. 데이타는 끊김없이 수신자에게 제공되어야 하므로 일정 시간 내에 데이타를 전송 받고 검증하지 못하면 그 데이타는 받지 못한 것과 마찬가지이다. 셋째, 스트림 데이타는 다양한 환경에서 다양한 경로를 통

† 비 회 원 : 서울대학교 대학원 전기컴퓨터 공학부
yspark@ssrnet.snu.ac.kr
†† 종신회원 : 서울대학교 컴퓨터학과 교수
cho@ssrnet.snu.ac.kr
논문접수 : 2002년 2월 20일
심사완료 : 2002년 6월 19일

해 전송되므로 경우에 따라서 많은 손실이 발생한다[7]. 따라서, 인증 기법은 데이타 손실에도 충분히 견딜 수 있어야 한다.

스트림 데이타의 인증에 관한 연구는 2 가지 접근 방식이 있으며(2장 참조), 본 논문에서 제안한 방식은 데이타를 그룹으로 묶은 후 서명 연산을 적용하여 많은 계산량을 요구하는 서명 연산의 횟수를 줄였다. 기존 방법 중 [2,11]은 이후의 논문에서 여러 가지 단점이 지적되었으며 [3,7], 현재, 이들 단점을 보완한 가장 최근 기법은 [3]의 기법과(이후 GM의 기법이라 부른다) [7]에서 제안한 EMSS(Efficient Multi-chained Stream Signature)와 확장된 EMSS가 있으며, 이들 기법은 모두 해쉬 체인 기법[1]을 사용한다.

본 논문에서는 Merkle이 제안한 인증 트리 기법[5]을 기반으로 한 효율적인 스트림 인증 기법을 제시한다. 동일 기법을 사용한 [11]과 비교해볼 때, 제안된 기법은 인증 트리 기법의 적용 방식이 다르다(2.2절 참조). 제안된 기법은 각 패킷의 인증 부가 정보에 인증 트리의 잎 노드부터 루트 노드까지의 경로 중 일정 높이까지만 포함시켜서 패킷의 크기를 줄였다. 또한, 연속된 패킷 손실에 대해 검증 효율을 높이기 위하여 패킷이 인증 트리의 잎 노드에 대응되는 순서를 설계하였으며, 계산량을 줄이기 위해 높이가 낮은 인증 트리를 고안하였다.

제안된 기법의 장점을 열거하면 다음과 같다. 첫째, 패킷 생성을 시뮬레이션하여 실험한 결과 기존 기법보다 검증 확률이 매우 뛰어났다. 일례로, 패킷 당 인증 부가 정보량이 40 바이트일 때, 제안된 기법의 검증 확률은 전 구간에서 87%이상이었으나, 기존 기법들은 패킷 손실률이 높아짐에 따라 9%-41%까지 낮아졌다. 둘째, EMSS, 확장된 EMSS와는 달리 수신된 데이타의 검증 지연 시간이 무한정 늦어지지 않으며, GM의 기법과는 달리 인증 부가 정보의 크기를 달리하여 원하는 검증 확률을 얻을 수 있다. 셋째, 우리는 제안된 기법의 검증 확률을 해석적으로 분석하였다. 넷째, 송신자가 필요한 계산량이 기존 기법 중 가장 적은 기법에 비견할 만하였으며, 각 기법을 구현하여 수행 속도를 측정한 결과 가장 빠른 GM의 기법보다 수행 속도가 약 0.7%~3.2% 증가하였다.

본 논문의 구성은 다음과 같다. 먼저 2 절에서 관련 연구를 서술한 후, 3 절에서 인증 트리 기법과 이를 이용한 제안된 기법을 제시한다. 그리고, 4 절에서는 제안된 기법과 기존 기법에 대하여 송신 서버의 계산량을 비교하고 제안된 기법의 검증 확률을 계산하며 실험을 통하여 각 기법의 검증 확률을 분석한 결과를 설명한다. 마지막으로, 5 절에서 결론을 맺는다.

1) 본 논문에서 해쉬 체인 기법은 한 패킷의 해쉬값을 다음 패킷에 저장하고, 저장된 패킷의 해쉬값을 그 다음 패킷에 저장하는 방식을 말하며 일반적인 해쉬 체인 기법과는 의미가 약간 다르다.

2. 관련 연구

스트림 데이타의 인증에 관한 연구는 크게 2 가지로 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 얻는다.

2.1. 효율적인 스트림 서명에 관한 연구

보다 효율적인 서명에 대한 연구는 여러 가지 방법으로 이루어져왔다. 간단한 예로, RSA 서명 기법에서 중국인의 나머지 정리(Chinese Remainder Theorem)나 슬라이딩 윈도우 기법을 사용하면 보다 빠른 서명 연산을 수행할 수 있다[4]. 또한 Wong과 Lam은 많은 메모리를 이용하여 Feige-Fiat-Shamir 서명 기법을 보다 효율적으로 구현할 수 있는 방법을 제시하였다[11]. 제시한 방법에서 서명 연산 속도와 메모리 사용량은 상충관계(trade-off)를 가지며, 최적화된 매개 변수를 선택하면 서명 연산을 DSA보다 2배 빠르게 수행하면서도 서명 확인 속도를 작은 지수 값을 가지는 RSA와 비슷한 수준으로 만들 수 있다. 또한, 서명을 확인할때 서명 확인자의 계산 능력치에 따라 서로 다른 수준의 신뢰를 주는 조정 가능하고 점진적인 서명 기법을 제안하였다.

또한, 효율적인 서명 기법으로 일회용 서명 기법이나 k 회용 서명 기법이 연구되었으며, Gennero와 Rohatgi는 이 기법을 이용하여 온라인 스트림 데이타에 대한 인증 기법을 제시하였다[2]. 다만, 이들 서명 기법은 일반적인 전자 서명 기법과는 달리, 서명의 크기가 서명될 데이타의 크기에 비례하며 SHA-1 해쉬 함수를 사용하는 경우 1300바이트가 넘는다. Rohatgi는 TCR(Target Collision Resistant) 해쉬 함수를 사용하는 등 여러 가지 방법을 이용하여 서명의 크기를 약 300 바이트로 줄였다[10].

2.2. 데이타를 그룹으로 묶는 방법에 대한 연구

데이타를 그룹으로 묶는 방법으로는 Gennaro와 Rohatgi가 제안한 방법으로 오프라인 방식이 있다[2]. 이 기법에서 패킷 그룹 내 k 번째 패킷은 k+1 번째 패

킷을 해쉬 함수에 입력한 결과값을 가지고 있으며 그룹 내 첫 번째 패킷만이 서명되어 전체적으로 서명의 횟수가 상당히 줄어든다(해쉬 체인 기법을 사용). 하지만, 이 기법은 패킷 데이타가 한 개라도 전송 도중 유실될 경우 그룹 내 그 이후의 패킷에 대해 검증할 수 없다는 단점을 가진다. 또한, 이 기법은 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.

Wong과 Lam은 Merkle이 고안한 인증 트리 기법[5]을 이용하여 데이타를 그룹으로 묶어 서명하였다[11]. 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. 본 논문에서 제안한 기법은 이 논문과 인증 트리 기법을 적용한 방식이 다르며, 1 절에서 언급하였듯이 높이가 낮은 트리를 고안하여 계산량을 줄였으며, 인증 트리 내 인증 정보를 패킷에 적절히 분배하여 패킷 크기를 상당히 줄였음에도 불구하고 수신자가 데이타를 검증시 검증 확률이 매우 높다(4.3절 참조).

Perrig, Canetti, Song, 그리고, Tygar는 TESLA (Timed Efficient Stream Loss-tolerant Authentication)와 EMSS라는 두 가지 기법을 제안하였다[7]. 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 번째 패킷의 해쉬값을 가지는 패킷들이 난수로 결정된다. 따라서 이 기법은 결정적(deterministic)이지 못하다. 특히, 이 기법은 받은 패킷이 검증되는 시점을 보장하지 못하며, 최대 무한대까지 지연될 수 있다. 이러한 결점을 보완하기 위하여, 저자들은 수신자가 허락하는 검증 지연 시간의 최대 한계값이 t라고 할 때, 시간 t동안 패킷들을 두 그룹 혹은 그 이상으로 나누어 패킷이 첫번째 그룹에서 검증되지 못하면 다음 그룹에서 검증되게끔 제안하였다. 이 방법을 사용하면 시간 t내 패킷의 검증 확률은 상당히 높아지지만 서명 횟수가 두 배 이상으로 늘어나서 계산량이 훨씬 늘어나게 된다. 같은 논문에서 저자는 IDA(Information Dispersal Algorithm) 기법[8]을 사용한 확장된 EMSS를 제안하였다. 이 기법은 검증 확률이 경우에 따라서 EMSS보다 높지만 역시 위에서 언급한 문제를 그대로 가지고 있으며 계산량 또한 EMSS보다 IDA 기법의 오버헤드만큼 많다.

Golle와 Modadugu는 일반적으로 인터넷에서 패킷 손실이 연속적으로 일어난다는 사실에 주목하였으며, 이를 고려한 GM의 기법을 제시하였다[3]. 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 검증 지연시간이 무한정 늦어지지 않으며, 송신 서버의 계산량이 매우 작다. 하지만, 이 기법은 인증 정보량을 조절할 수가 없어서 검증 확률을 원하는 만큼 얻을 수 없다.

3. 제안 기법

먼저, 3.1절에서 인증 트리 구조에 대해 설명하고 이를 기반으로 하여 3.2절에서 제안한 기법에 대해 설명한다. 본 논문에서 사용하는 용어의 의미는 다음과 같다. h(x)는 일방향 해쉬 함수이다. Sig(M)은 데이타 M을 전자 서명한 서명값이며, 서명자의 공개키를 이용하여 Sig(M)을 복호한 값과 h(M)이 일치하면 M이 성공적으로 검증된다. 또한, C||D는 데이타 C와 D를 연결(concatenate)시킨 것을 의미하며, e|f는 f가 e의 배수임을 뜻한다.

3.1 인증 트리

서명자(signer)와 검증자(verifier)가 존재하고, 서명자는 검증자에게 데이타 M_0, M_1, ..., M_{2^d-1}($d \geq 0$) 중 임의

의 $M_i(0 \le i < 2^d)$를 보내고 검증자는 받은 데이타를 검증하려고 한다. 이를 위해, 서명자는 M_0, M_1, ..., M_{2^d-1}에 대해 인증 트리를 구성한 후, 검증자에게 임의의 M_i와 인증 부가 정보(다음의 데이타 전송 부분에서 설명)를 보내며, 검증자는 이를 이용하여 M_i를 검증할 수 있다.

인증 정보 생성 인증 트리는 변형된 완전 이진 트리로써, 그림 1처럼 데이타 $M_i(0 \le i < 2^d)$에 대하여 각 잎 노드가 대응된다. 인증 트리의 각 노드 e에는 하나의 값 E가 연관되어 있다. e가 잎 노드인 경우, $e_{i\sim i}$로 표시하며 이에 연관된 값은 $E_{k\sim k}=h(M_k)$로 설정한다. e가 내부 노드인 경우, $e_{k\sim k'}(k<k')$로 표시하며 자식 노드 $e_{k1\sim k1'}$, $e_{k2\sim k2'}$, ..., $e_{kj\sim kj'}$, $(k_1 \le k_1' < k_2 \le k_2 < ... < k_j \le k_j')$를 가질 때 $k=k_1$, $k'=k_j'$로 설정된다. 또한, $e_{k\sim k'}$에 연관된 값은 $E_{k\sim k'}=h(E_{k1\sim k1'}||E_{k2\sim k2'}||...||E_{kj\sim kj'})$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 $Siblings(e_{1\sim1}, e_{0\sim3})=\{E_{0\sim0}, E_{2\sim3}\}$이다. 인증 트리는 루트 노드가 1개 존재하는 트리이면 어떤 트리 구조이든지 상관없으나[11], 설명의 용이를 위해 알고리즘 1에서는 완전 이진 트리(full binary tree)에 대한 인증 트리 생성 방법을 설명하며, 이진 트리가 아니거나 평형 트리가 아닌 경우 등의 인증 트리에 관한 자세한 내용은 [4,5,11]에 나와있다.

알고리즘 1 인증 트리 생성 알고리즘

```
1: 입력: 데이터 M_i (0 ≤ i < 2^d)
2: 출력: 인증 트리 A
3: for i = 0 TO i = 2^d − 1 do
4:     수준 d에 있는 잎 노드 e_{i~i}를 만든 후, E_{i~i} = h(M_i)를 연관시킨다.
5: end for
6: k = d − 1
7: while k ≥ 0 do
8:     for i = 0 TO i = 2^k − 1 do
9:         수준 k에 있는 내부 노드 e_{2^{d-k}i ~ 2^{d-k}i+2^{d-k-1}-1}을 만든다. 이 노드는 자식
           e_{2^{d-k}i ~ 2^{d-k}i+2^{d-k-1}-1}, e_{2^{d-k}i+2^{d-k-1} ~ 2^{d-k}(i+1)-1}을 가진다.
10:        E_{2^{d-k}i ~ 2^{d-k}(i+1)-1} = h(E_{2^{d-k}i ~ 2^{d-k}i+2^{d-k-1}-1} || E_{2^{d-k}i+2^{d-k-1} ~ 2^{d-k}(i+1)-1})을
           연관시킨다.
11:        k = k − 1
12:    end for
13: end while
```

데이타 전송 임의의 i에 대하여 서명자는 검증자에게 $(M_i, Sig(E_{0\sim2^d-1}), Siblings(e_{i\sim i}, e_{0\sim2^d-1}))$을 전송한다. 이 중 M_i는 검증자가 수신하여 검증해야할 데이타이며, 나머지는 검증에 필요한 인증 부가 정보이다.

데이타 검증 M_i를 검증하려면 인증 부가 정보를 이용하여 해당 잎 노드 $e_{i\sim i}$부터 루트 노드까지의 경로 상에 있는 각 노드에 연관된 값을 구한 후, 루트 노드와 연관된 값에 대한 서명값인 $Sig(E_{0\sim2^d-1})$을 검증해야하며 다음과 같은 절차를 따른다. 먼저, $Siblings(e_{i\sim i}, e_{0\sim2^d-1})=\{E_1, E_2, E_3, ..., E_k, ..., E_d\}$로 나타낼 수 있으며, E_k는 앞에서 언급한 경로 상에 있는 노드 중, 레벨 $k(1 \le k \le d)$에 있는 노드의 형제 노드에 연관된 값이다. 검증자는 경로 중 레벨 d에 있는 잎 노드 $e_{i\sim i}$에 연관된 값 $E'_{i\sim i}=h(M_i)$를 구한다. 경로 상의 노드 중 레벨 k-1에 있는 내부 노드에 연관된 값은 경로 상의 노드 중 레벨 k에 있는 노드에 연관된 값과 E_k를 이용하여 구할 수 있으며, k 값을 d부터 1까지 차례로 바꾸어 경로 상에 있는 각 노드에 연관된 값을 모두 구한다. 그 후, 송신자의 공개키를 이용하여 $Sig(E_{0\sim2^d-1})$을 복호한 값과 루트 노드와 연관된 값의 해쉬값인 $h(E'_{0\sim2^d-1})$이 일치하면 $E'_{0\sim2^d-1}$이 성공적으로 검증된 것이며 일방향 해쉬 함수의 성질에 따라 $E'_{0\sim2^d-1}$를 계산할 때 사용한 모든 값들이 검증된 것이고, 따라서 M_i가 성공적으로 검증된 것이다.

예 1. 그림 1은 잎 노드가 $4(=2^2)$개로 구성된 인증 트리를 보여주고 있다. 데이타는 M_0, M_1, M_2 그리고, M_3이며, 잎 노드에 연관된 값들은 각각 $E_{0\sim0}=h(M_0)$, $E_{1\sim1}=h(M_1)$, $E_{2\sim2}=h(M_2)$, 그리고 $E_{3\sim3}=h(M_3)$이다. 또한, 내부 노드와 연관된 값은 $E_{0\sim1}=h(E_{0\sim0}||E_{1\sim1})$, $E_{2\sim3}=h(E_{2\sim2}||E_{3\sim3})$, 그리고 $E_{0\sim3}=h(E_{0\sim1}||E_{2\sim3})$으로 계산된다. M_2를 검증하기 위해 필요한 인증 부가 정보는 $Sig(E_{0\sim3})$과 $Siblings(e_{2\sim2}, e_{0\sim3})=\{E_{3\sim3}, E_{0\sim1}\}$이며, M_2를 검증하는 방법은 다음과 같다. 먼저, $E'_{2\sim2}=h(M_2)$를 계산한 후 $E_{3\sim3}$을 이용하여 $E'_{2\sim3}=h(E'_{2\sim2}||E_{3\sim3})$을 계산한다. $E_{0\sim1}$과 $E'_{2\sim3}$을 가지고 $E'_{0\sim3}=h(E_{0\sim1}||E'_{2\sim3})$을 만들 수 있으며, $Sig(E_{0\sim3})$를 공개키로 복호화한 값과 $h(E'_{0\sim3})$이 같은지 확인하여 M_2를 검증한다.

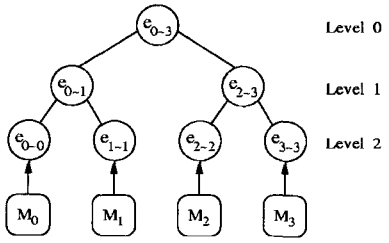


그림 1 인증 트리의 예

3.2 인증 트리 기법을 이용한 제안 기법

스트림 데이타 M_0, M_1, ..., $M_{tn-1}(n>0, t>0)$이 있어서 송신자는 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 부가 정보)를 생성하고 수신자에게 전송한다. 송신자는 모든 패킷 P_i를 전송한 후, 서명 패킷을 생성, 전송한다. 송신자는 위 작업을 $G_1=$

$\{M_{n+i}|(0\le i<n)\}$, $G_2=\{M_{2n+i}|(0\le i<n)\}$, ..., $G_{t-1}=\{M_{tn+i}|(0\le i<n)\}$에 대해 차례로 반복 수행한다. 수신자는 특정 그룹에 해당하는 패킷들과 서명 패킷을 받고 검증한다. 각 패킷에는 인증 부가 정보로 해쉬값이 포함되며 그 개수는 l이다. 또한, 서명 패킷은 n_s개의 해쉬값을 연결한 값과 이에 대한 서명값을 가지고 있다. 그리고, 송신자는 패킷 버퍼를 가지고 있어서 패킷을 한꺼번에 처리할 수 있으며, 그 크기는 2^b보다 같거나 크다고 가정한다. 제안된 기법에서 $n_s|n$, $2^l|(n/n_s)$이어야 하며, 본 절의 마지막 부분에서 이 조건을 만족시키는 n, n_s의 선택에 대하여 설명한다.

인증 트리 각 그룹마다 n_s개의 인증 트리가 있다. 인증 트리 $A_m(0\le m<n_s)$은 완전 이진 트리가 아니며, 잎 노드 수는 n/n_s이고, 루트 노드는 $n/(2^{l-1}n_s)$개의 완전 이진 서브 트리 $T_k(nm/(2^{l-1}n_s)\le k<n(m+1)/(2^{l-1}n_s))$를 가지며, 각 서브 트리는 잎 노드의 수가 2^{l-1}개이다(그림 2 참조). 또한, 잎 노드와 데이타가 연관되는 순서는 알고리즘 2의 4번째 줄에 따라 결정된다. 한편, 우리는 서브 트리의 쌍 $Pair_j=(T_{2j}, T_{2j+1})(nm/(2^ln_s)\le j<n(m+1)/(2^ln_s))$을 정의하며, 일례로 그림 2에서 $Pair_0=(T_0,T_1)$, $Pair_1=(T_2,T_3)$이다. 또한, 루트 노드와 연관된 값은 모든 서브 트리의 루트 노드와 연관된 값을 연결한 값의 해쉬값이다.

예 2. 그림 2는 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^l개인 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.

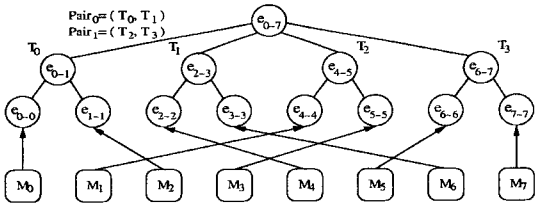


그림 2 제안 기법에서 사용하는 인증 트리의 예

패킷 생성 과정 송신자는 처음 n개의 스트림 데이타 $M_i(0\le i<n)$에 대하여(즉 G_0에 대하여), n_s개의 인증 트리 $A_m(0\le m<n_s)$을 다음과 같이 생성한다: 먼저, 알고리즘 2의 4번째 줄에 따라 n/n_s개의 스트림 데이타를 이용하여 인증 트리 A_m의 n/n_s개의 모든 잎 노드와 이에 연관된 값을 계산한 후, 알고리즘 1에 따라 A_m의 모든 완전 이진 서브 트리의 내부 노드와 이에 연관된 값을 계산한 다음, A_m의 루트 노드와 이에 연관된 값을 계산한다. 각 M_i마다 대응되는 하나의 잎 노드가 있으며(이를 $e_{c\sim c}$라 부른다.), $e_{c\sim c}$는 하나의 서브트리 T_a에 속해있고, T_a는 T_b(b=a+1 혹은 a-1)와 쌍을 이룬다. 송신자는 각 패킷 $P_i=(M_i, Siblings(e_{c\sim c}, T_a$의 루트 노드$)\cup\{T_b$의 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. 송신자는 위 작업을 G_1, G_2, ..., G_r, ..., G_{t-1}에 대해 반복 수행한다. 그룹 G_r에 대한 자세한 처리 방법이 알고리즘 2에 나와 있다.

예 3. l=2, n=32, n_s=4 그리고, 2^b=8 일 때, 송신자는 G_0에 대하여 예 2의 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_1은 T_0와 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.

알고리즘 2 제안 기법

1: **입력:** 스트림 데이터 M_i $(0\le i<n)$, l, 2^b, 그리고, n_s
2: **출력:** 패킷 P_i $(0\le i<n)$, 서명 패킷
3: **for** $m=0$ to n_s-1 **do**
4: 인증 트리 A_m 내 잎 노드 $e_{i\sim i}$는 $M_{(i \bmod 2^l)(2^{(b-l)})+\lfloor i/(2^l)\rfloor}$에 대응되며, $E_{i\sim i}=h(P_{i \bmod 2^l|(2^{(b-l)})+\lfloor i/(2^l)\rfloor})$이다.
5: **for** $k=\frac{n}{n_s2^{l-1}}m$ to $\frac{n}{n_s2^{l-1}}(m+1)-1$ **do**
6: 잎 노드 $\{e_{(2^{l-1})k\sim(2^{l-1})k}, e_{2^{l-1}k+1\sim2^{l-1}k+1}, \cdots, e_{2^{l-1}k+2^{l-1}-1\sim2^{l-1}k+2^{l-1}-1}\}$을 가지는 완전 이진 인증 서브 트리 T_k를 만든다.
7: **end for**
8: 서브트리의 쌍은 다음과 같이 정의된다: $Pair_j=(T_{2j}, T_{2j+1})$ $(\frac{n}{n_s2^l}m\le j<\frac{n}{n_s2^l}(m+1))$.
9: S_m 내 패킷 P_i를 생성한다. M_i가 서브 트리 T_a 내 잎 노드 $e_{c\sim c}$에 대응되고, T_a와 T_b가 한 쌍을 이룰 때, $P_i=(M_i, Siblings(e_{c\sim c}, T_a$의 루트 노드$)\cup\{T_b$의 루트 노드와 관련된 값$\})$이다.
10: A_m의 루트노드 $e_{\frac{n}{n_s}m\sim\frac{n}{n_s}(m+1)-1}$와 이에 연관된 값 $E_{\frac{n}{n_s}m\sim\frac{n}{n_s}(m+1)-1}=h(E_{\frac{n}{n_s}m\sim\frac{n}{n_s}m+2^{l-1}-1}||E_{\frac{n}{n_s}m+2^{l-1}\sim\frac{n}{n_s}m+(2)2^{l-1}-1}||E_{\frac{n}{n_s}m+(2)2^{l-1}\sim\frac{n}{n_s}m+(3)2^{l-1}-1}||\cdots||E_{\frac{n}{n_s}m+(\frac{n}{2^{l-1}n_s}-1)2^{l-1}\sim\frac{n}{n_s}m+(\frac{n}{2^{l-1}n_s})2^{l-1}-1})$을 구한다.
11: **end for**
12: 각 인증 트리의 루트노드와 연관된 값의 연결값을 구한다: $CData=A_0$의 루트 노드에 연관된 값$||A_1$의 루트 노드에 연관된 값$||\cdots||A_{n_s-1}$의 루트 노드에 연관된 값
13: 서명 패킷 $(CData, Sig(CData))$을 만든다.

패킷 검증 과정 수신자가 특정 그룹 $G_r(0\le r<t)$에 속하는 일부 패킷들과 서명 패킷을 수신하였을 때, 검증 과정은 인증 트리 A_m 단위로 이루어지며, 수신자는 각 A_m에 연관된 패킷들을 모두 검증하거나 검증하지 못한다. 수신자가 패킷 P_i를 수신하였으면, 패킷 내 인증 부가 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(전자의 값은 Siblings()를 이용하여 구할 수 있으며, 후자의 값은 인증 부가 정보에 들어 있다). 수신자가 A_m에 대한 각 서브 트리 쌍 $Pair_j$ $((2^l n_s)m \le j < n/(2^l n_s)(m+1))$의 잎노드에 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 송신자가 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. 우리는 알고리즘 2의 4번째 줄에 따라 연속된 스트림 데이타를 서브 트리 쌍마다 하나씩 대응시켜서 그러한 경우의 발생을 줄이려고 노력했다.

예 4. 예 3에서 수신자가 P_0, P_2, P_4, P_6 중 P_2 만을 받았다고 가정해보자. 이 패킷의 인증 부가 정보는 $\{E_{0\sim0}, E_{2\sim3}\}$이며, $E'_{1\sim1}=h(P_1)$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. 하지만 이 네 데이타 중 P_5만 전송 받았을 경우 이 패킷의 인증 부가 정보 $\{E_{7\sim7}, E_{4\sim5}\}$를 이용하여 $E'_{6\sim7}$을 계산할 수 있고 $E'_{0\sim7}$을 만들 수 있어서 서명 패킷을 수신했을 경우 P_2, P_5를 모두 검증할 수 있다.

제안된 기법은 $n_s|n$, $2^l|(n/n_s)$인 제약을 가진다. 일단, 그룹 크기를 $n_s|n'$, $2^l n_s|n'$를 만족시키는 $n'(<n)$으로 설정할 수 있으나 그룹의 크기가 작아져서 서명 연산 횟수가 늘어나게 되는 단점을 가진다. 또한, $n_s'|n$, $2^l|(n/n_s')$을 만족시키는 $n_s'(<n_s)$를 선택할 수 있다. 그리고, $n_s|n''$, $2^l n_s|n''$를 만족시키는 $n''(>n)$을 선택하고 $n''-n$개의 P_i들은 송/수신자가 미리 알고 있는 값으로 선택하되 이들 패킷은 전송하지 않는 방법도 있다.

4. 성능 분석

먼저, 4.1절에서 기존 기법과 제안된 기법에 대하여 송신 서버의 계산량을 비교하고, 4.2절에서 제안된 기법의 검증 확률을 해석적으로 분석하며, 4.3절에서 검증 확률에 대해 기존 기법과 비교 실험한 결과를 설명한다. 해쉬 함수와 전자 서명 알고리즘은 160비트 SHA-1, 1024비트 RSA를 선택했으며, 서명 패킷은 반복 전송 혹은 IDA 기법을 이용해서 손실 없이 보낸다고 가정한다. 또한, 비교 대상 기법은 GM의 기법, EMSS, 그리고 확장된 EMSS이다.

4.1 송신 서버의 계산량 비교

본 절에서는 제안된 기법과 기존 기법에 대해 송신 서버의 계산량을 비교한다. SHA-1 해쉬 함수의 계산량은 입력의 크기 x에 비례하며, $\alpha x+\beta$(α,β는 상수임)로 나타낼 수 있다[9]. 그리고, 서명 패킷을 생성하기 위한 전자 서명의 계산량은 γ로 정하며, 스트림 데이타 M_i의 크기는 m바이트, 패킷 생성률은 초당 r개, 검증 지연 시간의 한계값은 s 초라 하자. 1과 n_s는 3.2 절에서 언급하였으며, SHA-1 해쉬 함수를 사용하였으므로 패킷 당 인증 부가정보량은 20l 바이트이다.

제안된 기법에서 그룹의 크기는 최대 n=rs까지 설정할 수 있으며, 한 인증 트리 내 모든 잎 노드와 관련된 값을 생성하는데 필요한 계산량은 $(\alpha m+\beta)rs/n_s$이다. 한 인증 트리 내 완전 이진 서브 트리 쌍에 속하는 내부 노드는 모두 $(1/2+1/4+\ldots+1/2^{l-1})rs/n_s$개가 있으며, 이들 노드와 관련된 값들은 40 바이트에 대한 해쉬 결과이므로 이들을 계산하는데 필요한 계산량은$(1-1/2^{l-1})(rs/n_s)(40\alpha+\beta)$이다. 또한, 인증 트리의 루트 노드는 $rs/(n_s 2^{l-1})$개의 자식을 가지므로 루트 노드와 연관된 값을 구하는데 필요한 계산량은 $20rs/(n_s 2^{l-1})\alpha+\beta$이다. 따라서, 송신자가 한 그룹을 처리하는데 필요한 계산량은 $((m+40-20/(2^{l-1}))\alpha+(2-(1/2^{l-1})+n_s/(rs))\beta)rs+\gamma$이다. 우리는 EMSS와 GM의 기법의 계산량을 분석하였으며 표 1은 제시한 기법과 기존 기법의 계산량을 분석한 결과를 보인다. 단, 2절에서 언급하였듯이 EMSS에서 그룹의 크기는 rs/2로 설정하였다.

우리는 각 기법을 모두 구현하여 [7]에서 제시한 2가지 사례를 적용하여 송신 서버의 수행 시간을 측정하였다. 실험 환경은 다음과 같다. CPU와 메모리는 각각 Pentium III 650Mhz, 256Mbytes이고, 운영 체제, 암호 라이브러리, 컴파일러는 각각 Microsoft Windows 2000 Professional, Crypto^{++} 4.1, Microsoft Visual C^{++} 6.0이다. 실험 결과는 표 1에 있으며, 수행 시간은 rs개의 패킷에 대한 인증 정보를 생성하는 데 걸린 시간이고

표 1 송신 서버의 계산량 비교표

	계산량(rs개의 스트림 데이타를 처리하는 데 필요한 계산량)	수행 시간(ms)	
		사례 1	사례 2
EMSS	$((m+20\ell)\alpha+\beta)rs + 2\gamma$	30.26	48.14
GM의 기법	$((m+40)\alpha+\beta)rs + \gamma$	15.03	33.53
제안 기법	$((m+40-20/(2^{\ell-1}))\alpha+(2-(1/2^{\ell-1})+n_s/(rs))\beta)rs + \gamma$	15.15	34.63

10^4번 수행한 결과의 평균값이다. 본 논문에서 적용한 2가지 사례는 아래과 같다. 단, GM의 기법에서 각 패킷이 가지는 해쉬값의 개수가 평균 2개로 고정되므로, 모든 기법에서 l=2로 고정하였다.

· **사례 1: 교통 정보 데이타.** 스트림 데이타 M_i의 크기는 64byte이며 초당 20개가 전송된다. 패킷 손실률은 최대 5%이고, s=10초이며, n_s=4이다.

· **사례 2: 실시간 비디오 방송.** 스트림 데이타 M_i의 크기는 512byte이며 초당 512개가 전송된다. 패킷 손실률은 최대 60%이고, s=1초이며, n_s=40이다.

제안된 기법은 GM의 기법보다 수행시간이 약간 증가하였다. (사례 1: 0.7%, 사례 2: 3.2%). 두 기법에 대한 계산량을 분석한 식을 비교해 볼 때, 제안된 기법에서는 α의 계수가 GM의 기법의 그것보다 작지만, β의 계수는 더 크며 결과적으로 전체적인 계산량에서 큰 차이를 보이지 않는다. 이에 비해 EMSS의 수행 시간이 상당히 긴 원인은 그룹 크기가 rs/2이어서 서명에 대한 계산량이 두 배 많기(=2γ) 때문이다. 본 절에서는 확장된 EMSS의 계산량은 분석하지 않았으나, EMSS보다 IDA 기법의 오버헤드만큼 계산량이 더 많으므로, 제안된 기법이나 GM의 기법보다 계산량이 더 많아질 것으로 예상된다.

4.2 검증 확률에 대한 해석적 분석

본 절에서는 패킷 손실이 독립적으로 발생한다고 가정할 때 제안된 기법의 검증 확률을 해석적으로 분석한다. 먼저, 패킷 손실률을 1-p라고 가정하자.

정리 1. *제안된 기법의 검증 확률은*

$$P=\frac{E}{np}=(1-(1-p)^{2^l})^{\left(\frac{n}{2^l n_s}-1\right)}$$ *이다.*

증명. 패킷 P_k를 성공적으로 검증하기 위해서는 P_k에 연관된 잎 노드를 포함하는 인증 트리 A_m의 루트 노드와 연관된 값을 만들 수 있어야 한다. A_m의 잎 노드의 개수는 n/n_s개이며, A_m의 루트 노드는 $n/(2^{l-1}n_s)$개의 완전 이진 서브 트리의 루트 노드를 자식으로 가진다. 특정 서브 트리 쌍에 속하는 두 서브 트리의 잎 노드 수의 합을 $n_2=2^l$이라고 하고 A_m 내 서브 트리 쌍의 개수를 $n_1=n/(2^l n_s)$라 하면, A_m에 연관된 n_1n_2개의 패킷 중, 수신자가 수신할 수 있는 패킷 개수의 기대값은 $E_0=\sum_{i=0}^{n_1n_2} i\,({}_{n_1n_2}C_i)p^i(1-p)^{n_1n_2-i}=n_1n_2p$이다.

하지만, 특정 서브 트리 쌍 $Pair_j$에 속하는 모든 n_2개의 잎 노드들이 전송 도중 손실되었을 경우를 생각해보자. 그때는 $Pair_j$를 구성하는 두 서브 트리의 루트 노드를 계산할 수 없으며, 따라서 A_m에 연관된 모든 패킷에 대한 검증이 실패하게 된다. 수신하여 검증 가능한 패킷 개수의 기대값은 E_0에서 이런 경우들을 모두 제외한 값이다.

먼저, 첫 번째 서브 트리 쌍에 해당하는 패킷이 모두 손실된 경우를 E_0에서 제외하면 $n_1n_2p-(1-p)^{n_2}\sum_{i=0}^{n_1n_2-n_2} i\,({}_{(n_1n_2-n_2)}C_i)p^i(1-p)^{n_1n_2-n_2-i}=n_1n_2p-(1-p)^{n_2}(n_1n_2-n_2)p$가 된다. 서브 트리의 쌍은 n_1개 있으므로 두 번째, 세 번째, ..., n_1 번째 서브 트리 쌍이 손실된 경우를 전부 제외하면 $E_1=n_1n_2p-({}_{n_1}C_1)(1-p)^{n_2}(n_1n_2-n_2)p$이다. 하지만, 이 값에는 해당 패킷이 모두 손실된 트리 쌍이 동시에 2 번 이상 생길 경우에 대한 계산이 중복되어 있다. 먼저, 2번 생길 경우를 가만하면 $E_2=n_1n_2p-({}_{n_1}C_1)(1-p)^{n_2}(n_1n_2-n_2)p+({}_{n_1}C_2)(1-p)^{2n_2}(n_1n_2-2n_2)p$로 생각해 볼 수 있으나, 이 식에는 동시에 3 번 이상 생길 경우에 대한 계산이 중복으로 들어 있다. n_1 번까지 생길 경우를 고려하면, 식 $E_{n_1}=\sum_{i=0}^{n_1}(-1)^i({}_{n_1}C_i)(1-p)^{n_2i}(n_1n_2-n_2i)p$을 얻을 수 있으며, 이항 정리 공식을 적용하면 $E_{n_1}=(n_1n_2p)(1-(1-p)^{n_2})^{(n_1-1)}$ 이 된다. E_{n_1}은 하나의 인증 트리에 대하여 계산한 값이며, 전체 그룹에 대한 기대값은 $E=n_sE_{n_1}$이고 검증 확률은

$$P=\frac{E}{np}=(1-(1-p)^{2^l})^{\left(\frac{n}{2^l n_s}-1\right)}$$ 이다.

위 결과는 패킷 손실을 베르누이 모델에 따라 독립적으로 발생시켜서 얻은 실험 결과와 거의 일치하였다. 하지만, 2-상태 마르코프 모델에 따라 패킷 손실을 발생시켰을 때(4.3절 참조), 실험 결과가 해석식보다 높은 검증 확률을 보였는데 그 이유는 다음과 같이 추정된다: 2-상태 마르코프 모델에 기반한 실험에서는 패킷 손실이 연속으로 일어나며, 성공적인 수신 또한 연속적으로 일어난다. 3.2 절에서 언급했듯이 알고리즘 2의 4번째 줄의 매핑에 의하여, 연속적으로 수신된 패킷이 각 서브 트리 쌍의 잎 노드 중 하나에 대응되며, 일례로 M_0, M_1, M_2, ..., M_k는 각각 e_{0*0}, e_{1*2^l}, e_{2*2^l}, ..., e_{k*2^l} ($2^lk<2^b$)에 대응된다. 따라서, 같은 서브 트리에 해당하는 모든 패킷이 손실되는 경우가 상당히 줄어들며, 패킷 손실을 독립적으로 발생시킨 경우보다 검증 확률이 높아질 가능성이 크다.

4.3 실험 결과

본 절에서는 시뮬레이션을 통해 각 기법들의 검증 확률을 분석한 결과를 설명한다. 제안된 기법과 GM의 기법에서는 n=1024로 설정하였다. EMSS와 확장된 EMSS에서는 n=512로 설정하되 수신자가 특정 패킷

P_i를 검증 시, P_i가 속한 그룹과 다음 그룹에 대한 패킷들의 인증 부가 정보를 이용하여 P_i가 검증되지 않으면 P_i는 검증 실패로 간주하였다. 또한, 각 기법에서 n_s=32, 2^b=64로 설정하였다. 패킷 손실은 2-상태 마르코프 모델에 따라 발생시켰다[7,12]. 이전 패킷이 손실되었을 경우 다음 패킷이 손실될 확률은 이전 패킷을 받았을 경우 다음 패킷이 손실될 확률보다 7 배 높게 설정하였으며, 이 값은 인터넷 상의 패킷 손실에 대하여 [1,6]에서 분석한 결과이다. 실험 결과는 10^5 번을 수행하여 나온 결과의 평균값이다.

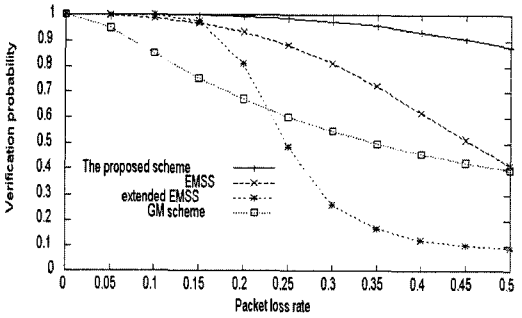


그림 3 실험 결과 1

그림 3은 패킷 당 인증 부가 정보량이 40바이트일 때, 각 기법의 검증 확률을 보여준다. 가로 축은 패킷 손실률이며 세로 축은 검증 확률이다. 제안된 기법은 기존 기법들보다 검증 확률이 상당히 높으며, 패킷 손실률이 높아질수록 현저한 차이를 보였다. GM의 기법은 다른 기법들에 비해 검증 확률이 현저히 낮았다. EMSS는 제안된 기법보다 검증 확률이 낮았으며, 패킷 손실률이 25% 이후부터 검증 확률이 상당히 낮아짐을 보였다. 확장된 EMSS는 패킷 손실률이 0~15% 구간에서는 제안된 기법보다는 낮고 EMSS보다는 높았지만, 그 이후의 구간에서는 검증 확률이 상당히 낮았다.

일례로, 패킷 손실률이 15%이면, 1024개의 패킷을 전송했을 때 수신된 패킷 수가 평균 870.4개이다. 이 때, 수신되었으나 검증이 실패할 패킷의 수를 비교해보면 새 기법은 평균 12.4개로 가장 적다. 반면, EMSS는 29.5개, 확장된 EMSS는 22.7개이며, GM의 기법은 213.5개나 되어 수신된 패킷의 약 25%가 검증되지 못한다.

그림 4는 인증 부가 정보량이 60바이트일 때 각 기법의 검증 확률을 보여준다. 단, GM의 기법에서는 각 패킷이 가지는 해쉬값의 개수가 평균 2개로 고정되므로

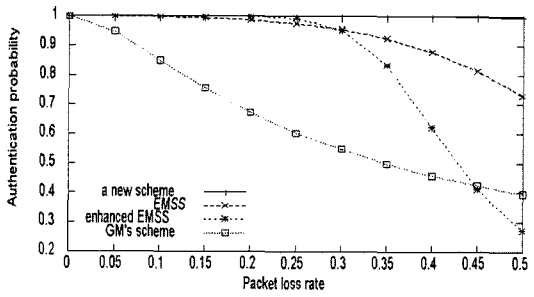


그림 4 실험 결과 2

인증 부가 정보량을 40바이트로 설정하였다. 그림 3과 비교해볼 때, 인증 부가 정보량이 늘어남에 따라 전체적인 검증 확률이 늘어났다. 제안된 기법의 검증 확률은 EMSS, 확장된 EMSS, 그리고 GM의 기법보다 훨씬 높을 뿐만 아니라, 패킷 손실률이 심지어 50%일 때도 검증 확률이 약 99.5%나 되어 거의 모든 패킷을 검증할 수 있음을 보였다.

5. 결 론

본 논문에서 인증 트리 기법을 이용하여 실시간으로 전달되는 스트림 데이타를 인증하는 효율적인 기법을 제안하였다. 제안된 기법은 송신자가 매우 적은 계산량으로 인증 정보를 생성할 수 있고 수신자가 받은 데이타를 검증할 때 검증 확률이 매우 높다. 본 논문에서는 제안된 기법의 검증 확률을 해석적으로 분석한 결과를 제시하였다. 기존 기법들과 비교해보면 첫째, 송신 서버의 계산량은 기존 기법 중 가장 우수한 GM의 기법과 거의 비슷한 수준이며, EMSS와 확장된 EMSS보다는 상당히 많은 계산량의 감소를 가져왔다. 둘째, 시뮬레이션을 통해 검증 확률을 측정한 결과, GM의 기법은 검증 확률이 타 기법보다 현저히 낮으며, 제안된 기법은 기존 기법들보다 매우 높은 검증 확률을 보였다. 일례로, 패킷 당 인증 부가 정보량이 40바이트일 때, 제안된 기법의 검증 확률은 전 구간에서 87% 이상이었으나, 기존 기법들은 패킷 손실률이 높아짐에 따라 9%~41%까지 낮아졌다. 셋째, 제안된 기법은 EMSS, 확장된 EMSS와는 달리 수신된 데이타의 검증 지연 시간이 무한정 늦어지지 않으며, GM의 기법과는 달리 인증 부가 정보의 크기를 달리하여 원하는 검증 확률을 얻을 수 있다.

참 고 문 헌

[1] Michael S. Borella, Debbie Swider, S. Uludag, and G. Brewster. Internet Packet Loss: Measurement and Implications for End-to-End QoS, In International Conference on Parallel Processing, 1998.

[2] Rosario Gennaro and Pankaj Rohatgi, How to Sign Digital Streams, In CRYPTO'97, pages 180-197, 1997.

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

[4] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone, Handbook of Applied Cryptography, CRC Press, 1997.

[5] Ralph C. Merkle, A Certified Digital Signature, In CRYPTO'89, pages 218-238, 1989

[6] Vern Paxson, End-to-End Internet Packet Dynamics, IEEE/ACM Transactions on Networking, 7(3):277-292, 1999.

[7] 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.

[8] 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.

[9] Michael Roe, Performance of Protocols, In Security Protocols Workshop, LNCS vol. 1796, pp. 140-146, 1999.

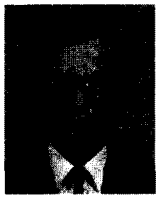
[10] Pankaj Rohatgi, A Compact and Fast Hybrid Signature Scheme for Multicast Packet Authentication, In 6th ACM Conference on Computer and Communication Security, pp. 93-100, November, 1999.

[11] Chung Kei Wong and Simon S. Lam, Digital Signatures for Flows and Multicasts, IEEE/ACM Transactions on Networking, 7(4):502-513, 1999.

[12] M. Yajnik, S. Moon, J. Kurose, and D. Towsley, Measurement and modelling of the temporal dependence in packet loss, In IEEE INFOCOM'99, 1999.

박 용 수

1992년 ~ 1996년 한국과학기술원 전산학과(학사). 1996년 ~ 1998년 서울대학교 컴퓨터공학과(석사). 1998년 ~ 현재 서울대학교 컴퓨터공학과 박사과정. 관심분야는 정보보안, 네트워크보안, 암호학



조 유 근

1971년 서울대학교 건축공학과 학사. 1978년 미네소타대학교 전산학 박사. 1979년 ~ 현재 서울대학교 컴퓨터공학부 교수. 1984년 ~ 1985년 미네소타대학교 교환 교수. 1993년 ~ 1995년 서울대학교 중앙교육연구전산원장. 1995년 한국정보과학회 부회장. 1999년 ~ 2001년 서울대학교 공과대학 부학장. 2001년 ~ 현재 한국정보과학회 회장. 관심분야는 운영체제, 알고리즘 설계 및 분석, 암호학