

## 정보 분산 알고리즘을 이용한 PCC 기법의 개선

현상원, 박용수, 조유근

서울대학교, 전기.컴퓨터 공학부

### An Improvement of PCC Scheme by Using Information Dispersal Algorithm

Sang-weon Hyun, Yong-su Park, Yoo-kun Cho

School of Computer Science and Engineering, Seoul National Univ.

#### 요약

본 논문에서는 최근에 발표된 스트림 인증 기법중 하나인 PCC 기법 [1]을 개선한 방법을 제시한다. PCC 기법에서는 수신된 스트림 데이터의 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 SAIDA보다 검증 확률이 높고, 구현 프로그램의 수행 시간을 측정할 결과 계산 시간 면에서 더 나은 성능을 보였다.

#### I. 서론

##### 1. 연구배경

인터넷 사용자가 늘어나고 망 대역폭이 커짐에 따라 인터넷을 통하여 오디오나 비디오 같은 스트림 데이터를 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 데이터 무결성 (data integrity)을 제공하는 인증 기법이 무엇보다 중요하다.

실시간으로 생성/전달되는 스트림 데이터를 인증하는 문제는 일반 데이터를 인증하는 것과 다르며, 다음과 같은 문제점을 해결해야한다. 첫째, 서

명 연산량을 최소화해야 한다. 전자 서명이 많은 계산량을 요구하는 작업을 생각해보면, 모든 데이터를 서명하는 것은 송신 서버에게 상당한 부하를 준다. 둘째, 통신 오버헤드의 크기를 최소화해야 한다 [4]. 셋째, 스트림 데이터는 다양한 환경에서 다양한 경로를 통해 전송되므로 경우에 따라서 많은 손실이 발생한다 [2]. 따라서, 인증 기법은 데이터 손실에도 충분히 높은 검증 확률 (수신된 패킷들 중 검증 가능한 패킷 수의 비 [4])을 제공해야 한다.

본 논문에서 제안하는 기법은 PCC 기법 [1]에 기반하며 Rabin의 정보 분산 알고리즘 (Information Dispersal Algorithm) [3]을 이용해서 PCC 기법의 약점을 다음과 같이 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 [2], Piggybacking 기법 [6]과는 달리 서명 패킷을 가지지 않는다. 둘째, 패킷 생성을 시뮬레이션하여 실험한 결과 PCC 기법과 관련 최신 기법인 SAIDA 기법 [4]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.

## 2. 표기법

본 논문에서 사용하는 용어의 의미는 다음과 같다.  $h(x)$ 는 일방향 해쉬 함수이다.  $Sig_A(M)$ 은 데이터  $M$ 을 서명자  $A$ 의 개인키로 전자 서명한 서명 값이며,  $A$ 의 공개키를 이용하여  $Sig_A(M)$ 을 복호한 값과  $h(M)$ 이 일치하면  $M$ 이 성공적으로 검증된다. 본 논문에서 서명자는 스트림 데이터를 전송하는 송신 서버이며  $SIG(M)$ 은  $M$ 에 대한 송신 서버의 서명을 뜻한다. 또한,  $C||D$ 는 데이터  $C$ 와  $D$ 를 연결(concatenate)시킨 것,  $|E|$ 는 데이터  $E$ 의 바이트 단위 크기를 그리고,  $elf$ 는  $f$ 가  $e$ 의 배수임을 뜻한다.

## II. 본문

### 1. PCC(Park Chung Cho) 기법 [1]

스트림 데이터  $M_0, M_1, \dots, M_{t-1}$  ( $n > 0, t > 0$ )이 있어서 송신 서버는 처음  $n$ 개의  $M_i$  ( $0 \leq i < n$ )에 대해 (이들을 그룹  $G_0$ 라 부른다) 인증 정보를 생성한다.  $G_0$  내 각  $M_i$ 에 대하여 패킷  $P_i = (M_i, \text{인증 정보})$ 를 생성하고 수신자에게 전송한다. 송신 서버는 모든 패킷  $P_i$ 를 전송한 후, 서명 패킷을 생성, 전송한다. 송신 서버는 위 작업을  $G_1 = \{M_{n+i} | (0 \leq i < n)\}$ ,  $G_2 = \{M_{2n+i} | (0 \leq i < n)\}$ , ...,  $G_{t-1} = \{M_{(t-1)n+i} | (0 \leq i < n)\}$ 에 대해 차례로 반복 수행한다. 수신자는 특정 그룹에 해당하는 패킷들과 서명 패킷을 받고 검증한다. 각 패킷에는 인증 정보로 해쉬값이 포함되며 그 개수는  $l$ 이다. 또한, 서명 패킷은  $n_s$ 개의 해쉬값을 연결한 값과 이에 대한 서명값을 가지고 있다. 그리고, 송신 서버는 패킷 버퍼를 가지고 있어서 패킷을 한꺼번에 처리할 수 있으며, 그 크기는  $2^b$ 보다 같거나 크다고 가정한다. PCC 기법

에서는  $n_s | n, 2^l | (n/n_s)$ 이어야 하며, 이 조건을 만족시키는  $n, n_s$ 의 선택에 대한 자세한 내용은 [1]에 나와 있다.

인증 트리 각 그룹마다  $n_s$ 개의 인증 트리가 있다. 인증 트리  $A_m$  ( $0 \leq m < n_s$ )의 잎 노드 수는  $n/n_s$ 이다.  $A_m$ 의 루트 노드는  $n/(2^{l-1}n_s)$ 개의 완전 이진 서브 트리  $T_k$  ( $nm/(2^{l-1}n_s) \leq k < n(m+1)/(2^{l-1}n_s)$ )의 루트 노드를 자식으로 가지며, 각 서브 트리  $T_k$ 는 잎 노드의 수가  $2^{l-1}$ 개이다 (그림 1 참조).  $A_m$ 의 각 노드  $e$ 에는 하나의 값  $E$ 가 연관되어 있으며, 각  $M_i$ 마다 하나의 잎 노드  $e_{c \sim c'}$ 가 대응된다. 또 이에 연관된 값은  $E_{c \sim c'} = h(M_i)$ 로 설정한다. 또  $e_{c \sim c'}$ 는 하나의 서브트리  $T_a$ 에 속해있고,  $T_a$ 는  $T_b$  ( $b = a+1$  혹은  $a-1$ )와 쌍을 이룬다.  $e$ 가 내부 노드인 경우,  $e_{c \sim c'}$  ( $c < c'$ )로 표시하며 자식 노드  $e_{c_1 \sim c'_1}, e_{c_2 \sim c'_2}, \dots, e_{c_l \sim c'_l}$  ( $c_1 \leq c'_1 < c_2 \leq c'_2 < \dots < c_l \leq c'_l$ )를 가질 때  $c = c_1, c' = c'_1$ 로 설정된다. 또한,  $e_{c \sim c'}$ 에 연관된 값은  $E_{c \sim c'} = h(E_{c_1 \sim c'_1} || E_{c_2 \sim c'_2} || \dots || E_{c_l \sim c'_l})$ 이며, 일례로, 그림 1에서  $E_{0 \sim 1} = h(E_{0 \sim 0} || E_{1 \sim 1})$ 이다. 우리는 노드  $e$ 에서 조상 노드  $e'$ 까지 경로 상에 있는 각 노드의 모든 형제 노드와 연관된 값의 집합을 Siblings( $e, e'$ )라고 부르며, 일례로, 그림 1에서 Siblings( $e_{0 \sim 0}, e_{0 \sim 1}$ ) =  $\{E_{1 \sim 1}\}$ 이다. 서브 트리의 쌍  $Pair_j = (T_{2j}, T_{2j+1})$  ( $nm/(2^l n_s) \leq j < n(m+1)/(2^l n_s)$ )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  $T_k$ 의 루트 노드와 연관된 값을 연결한 값의 해쉬값이다. 또한, 패킷 손실이 연속적으로 발생하는 특성을 띠는 인터넷 환경에서 높은 검증 확률을 제공하기 위해 인증 트리의  $i$ 번째 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,  $G_0$ 에 대한 인증 트리 4개 중 하나인  $A_0$ 의 구조를 보여주고 있다. 인증 트리의 루트 노드  $e_{0 \sim 7}$ 은 잎 노드가  $2^{l-1}$ 개인 완전 이진 서브 트리  $T_0, T_1, T_2, T_3$ 의 루트 노드를 자식으로 가지며, 루트 노드와 연관된 값  $E_{0 \sim 7}$ 은  $h(E_{0 \sim 1} || E_{2 \sim 3} || E_{4 \sim 5} || E_{6 \sim 7})$ 이다.

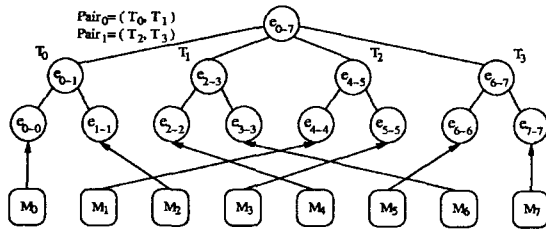


그림 1: PCC 기법에서 사용하는 인증 트리의 예

**패킷 생성 과정** 송신 서버는 처음  $n$ 개의 스트림 데이터  $M_i$  ( $0 \leq i < n$ )에 대해 (즉  $G_0$ 에 대해),  $n_s$ 개의 인증 트리  $A_m$  ( $0 \leq m < n_s$ )을 다음과 같이 생성한다: 먼저,  $n/n_s$ 개의 스트림 데이터를 이용하여 인증 트리  $A_m$ 의 모든 잎 노드와 이에 연관된 값을 계산한 후,  $A_m$ 의 모든 완전 이진 서브 트리의 내부 노드와 이에 연관된 값을 계산한 다음,  $A_m$ 의 루트 노드와 이에 연관된 값을 계산한다. 송신 서버는 각 패킷  $P_i = (M_i, \text{Siblings}(e_{c-c}, T_a \text{의 루트 노드}) \cup \{T_b \text{의 루트 노드와 연관된 값}\})$ 를 생성하며 수신자에게 전송한다. 모든  $A_m$  ( $0 \leq m < n_s$ )을 만들고 이에 대응하는 패킷들을 전송한 후, 서명 패킷을 생성, 전송한다. 단, 서명 패킷의 내용은 모든  $A_m$  ( $0 \leq m < n_s$ )의 루트 노드에 연관된 값을 연결한 값과 이를 서명한 값이다. 송신 서버는 위 작업을  $G_1, G_2, \dots, G_{t-1}$ 에 대해 반복 수행한다.

**패킷 검증 과정** 서명 패킷 손실 시에는 수신된 모든 패킷의 검증이 불가능하며, 서명 패킷을 성공적으로 수신했을 경우의 검증 과정은 다음과 같다. 수신자가 특정 그룹  $G_r$  ( $0 \leq r < t$ )에 속하는 일부 패킷들을 수신하였을 때, 검증 과정은 인증 트리  $A_m$  단위로 이루어지며, 수신자는 각  $A_m$ 에 연관된 패킷들을 모두 검증하거나 검증하지 못한다. 수신자가 패킷  $P_i$ 를 수신하면, 패킷 내 인증 정보를 이용하여  $M_i$ 에 대응하는 잎 노드  $e_{c-c}$ 를 포함하는 서브 트리  $T_a$ 의 루트 노드와 연관된 값과,  $T_a$ 와 같은 서브 트리 쌍에 속하는 트리  $T_b$ 의 루트 노드와 연관된 값을 구할 수 있다. 수신자가  $A_m$ 에 대한 각 서브 트리 쌍  $\text{Pair}_j$  ( $((2^l n_s) m \leq j < n / (2^l n_s) (m+1))$ )의 잎 노드에 대응하는  $2^l$ 개의 패킷 중 적어도 1개의 패킷을 수신했으면,  $A_m$ 의 모든 서브 트리의 루트 노드와 연관된 값을 계산할 수 있으며, 따라서  $A_m$ 의 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  $T_k$ 의 루트 노드와 관련된 값이 검증된 것이고,  $A_m$ 의 잎 노드에 대응되는 각 패킷들도 성공적으로 검증된 것이다. 하지만, 특정 서브

트리 쌍에 대응하는  $2^l$ 개의 패킷이 모두 손실된 경우  $A_m$ 의 루트 노드에 연관된 값을 구할 수 없다. 이러한 경우의 발생을 줄이기 위해 식  $f(i) = (i \bmod 2^l)(2^b - 1) + \lfloor i/2^l \rfloor$ 에 의해 연속된 스트림 데이터를 서브 트리 쌍마다 하나씩 대응되게 했다.

## 2. 제안 기법

PCC 기법은 서명 패킷이 손실될 경우 수신된 모든 패킷의 검증이 불가능하다는 문제점을 갖는다. 이에 [1,2]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 알고리즘을 사용한다.

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

**패킷 생성 과정** 송신 서버는 처음  $n$ 개의 스트림 데이터  $M_i$  ( $0 \leq i < n$ )에 대해 (즉  $G_0$ 에 대해),  $n_s$ 개의 인증 트리  $A_m$  ( $0 \leq m < n_s$ )을 PCC 기법에 의해 생성한다. 모든  $A_m$  ( $0 \leq m < n_s$ )이 만들어지고 나면 서명 패킷  $\text{SIG} = (E_{0-n/n_s-1} || E_{n/n_s-2n/n_s-1} || \dots || E_{(n_s-1)n/n_s-n-1}, \text{SIG}(E_{0-n/n_s-1} || E_{n/n_s-2n/n_s-1} || \dots || E_{(n_s-1)n/n_s-n-1}))$ 를 생성한다. 생성된 서명 패킷에 대해서 정보 분산 연산  $\text{Dispersal}(\text{SIG}, m, n)$ 을 적용하며, 그 결과는 서명 패킷에 대한  $n$ 개의 단편들  $\text{SIG}_i$  ( $0 \leq i < n$ )가 된다. 송신 서버는 각 패킷  $P_i = (M_i, \text{Siblings}(e_{c-c}, T_a \text{의 루트 노드}) \cup \{T_b \text{의 루트 노드와 연관된 값}\}, \text{SIG}_i)$ 를 생성하며 수신자에게 전송한다. 따라서 송신 서버에 필요한 패킷 버퍼의 크기  $2^b = n$ 이다. 송신 서버는 위 작업을  $G_1, G_2, \dots, G_{t-1}$ 에 대해 반복 수행한다.

**예 2.**  $l=2, n=32, n_s=4, m'=16$  그리고  $2^b=n$ 일 때의 패킷 생성 과정은 다음과 같다. 해쉬 함수와 전자 서명 알고리즘으로 [4]에서 사용한 128비트 MD5, 1024비트 RSA를 선택하면,  $|h()|=16$ ,

$|SIG|=128$ 이다. 송신 서버는  $G_0$ 에 대하여 예 1의 인증 트리  $A_0$ 를 포함한 4개의 인증 트리를 생성한 후, 32개의 패킷  $P_i$  ( $0 \leq i < 32$ )를 생성한다. 그 후, 서명 패킷  $SIG=(E_{0-7}||E_{8-15}||E_{16-23}||E_{24-31}, SIG(E_{0-7}||E_{8-15}||E_{16-23}||E_{24-31}))$  ( $|SIG|=192$ 바이트)를 생성하고 정보 분산 연산을 적용한다. 그 결과는 32개의 단편들  $SIG_i$  ( $0 \leq i < 32$ )가 되며, 각 단편의 크기는 12바이트( $=192/16$ )가 된다. 그림 2는 인증 트리  $A_0$ 에 대응되는 패킷들의 생성 예이다. 일례로, 전송되는 패킷들 중 하나인  $P_8$ 은 ( $M_8, \{E_{0-0}, E_{2-3}\}$ ,  $SIG_8$ )가 된다.

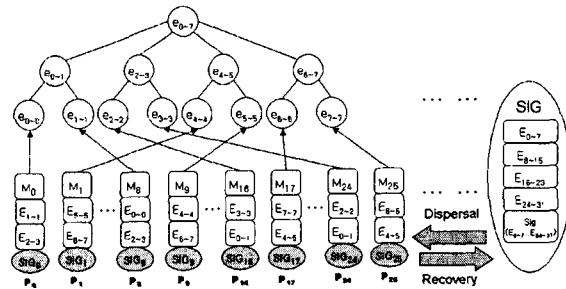


그림 2: 제안 기법에서 패킷 생성 및 검증 예

패킷 검증 과정 수신자가 특정 그룹  $G_r$  ( $0 \leq r < t$ )에 속하는 전체  $n$ 개의 패킷들 중  $m'$ 개 미만의 패킷만을 수신할 경우에는 정보 복원 연산을 통한 서명 패킷의 복원이 불가능하므로 검증이 실패한다. 수신자가  $n$ 개의 패킷들 중  $m'$ 개 이상의 패킷들을 수신했을 경우에는 정보 복원 연산  $Recovery(\frac{1}{2}SIG_i | (0 \leq j < m0), (0 \leq i < n) \frac{1}{2}, m0, n)$ 을 통해 서명 패킷이 복원되며, 이후의 검증 과정은 PCC 기법과 동일하다. 수신자가 인증 트리  $A_m$ 에 대한 각 서브 트리 쌍  $Pair_j$  ( $(2'n_s)m \leq j < n/(2'n_s)(m+1)$ )의 잎 노드에 대응하는  $2'$ 개의 패킷 중 적어도 1개의 패킷을 수신했으면,  $A_m$ 의 모든 서브 트리의 루트 노드와 연관된 값을 계산할 수 있으며, 따라서  $A_m$ 의 루트 노드에 연관된 값을 계산할 수 있다. 이 값이 복원된 서명 패킷 내 송신 서버가 만든 값과 일치하면 성공적으로 검증된 것이며, 일방향 함수의 성질에 따라 이 값을 계산할 때 사용한 모든  $T_k$ 의 루트 노드와 관련된 값이 검증된 것이고,  $A_m$ 의 잎 노드에 대응되는 각 패킷들도 성공적으로 검증된 것이다. 하지만, 특정 서브 트리 쌍에 대응하는  $2'$ 개의 패킷이 모두 손실된 경우  $A_m$ 의 루트 노드에 연관된 값을 구할 수 없으므로 검증이 실패한다.

예 3. 예 2에서 수신자가  $G_0$ 에 속한 32개의 패킷들 중 16개 이상의 패킷들을 수신했으며, 수신된 패킷에  $P_0, P_8, P_{16}, P_{24}$  중  $P_8$ 만 포함되어 있다

고 가정하자. 수신자는 우선 16개의 정보 분산 연산 단편에  $Recovery(\frac{1}{2}SIG_i | (0 \leq j < 16), (0 \leq i < 32) \frac{1}{2}, 16, 32)$ 을 적용해서 서명 패킷을 복원한다.  $P_8$  패킷의 인증 부가 정보는 ( $E_{0-0}, E_{2-3}$ )이며,  $E'_{1-1}=h(M_8)$ 을 계산하여  $E'_{0-1}=h(E_{0-0}||E'_{1-1})$ 을 계산할 수 있다. 이 때,  $P_1, P_9, P_{17}, P_{25}$ 가 모두 손실된 경우,  $E_{4-5}, E_{6-7}$ 을 구할 수 없어서  $P_8$ 을 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 ( $E_{7-7}, E_{4-5}$ )이며,  $E'_{6-6}=h(M_{17})$ 을 계산하여  $E'_{6-7}=h(E'_{6-6}||E_{7-7})$ 을 계산할 수 있다. 최종적으로  $E'_{0-7}=h(E'_{0-1}||E_{2-3}||E_{4-5}||E'_{6-7})$ 을 계산할 수 있으며,  $E'_{0-7}$ 와 복원된 서명 패킷에 포함된  $E_{0-7}$ 을 비교함으로써 수신된 패킷  $P_8, P_{17}$ 을 모두 검증할 수 있다.

### 3. 성능 분석

해쉬 함수와 전자 서명은 [4]에서 사용한 알고리즘인 128비트 MD5, 1024비트 RSA를 선택했으며, 비교 대상 기법은 PCC 기법과 SAIDA 기법이 다.

#### 1) 실험 결과

본 절에서는 시뮬레이션을 통해 각 기법들의 검증 확률을 분석한 결과를 설명한다. PCC 기법의 경우 두개의 서명 패킷을 128개의 패킷만큼 지연 후 반복 전송했다. 손실이 연속적으로 발생하는 인터넷의 전송 특성을 고려하여 패킷 손실은 2-상태 마르코프 모델에 따라 발생시켰다 [1,4]. [4]에 따라 그룹의 크기  $n=128$ , 패킷 손실을 20%, 연속적으로 손실된 패킷 수의 평균값을 8로 설정했다. 실험 결과는  $10^5$  번을 수행하여 나온 결과의 평균 값이다.

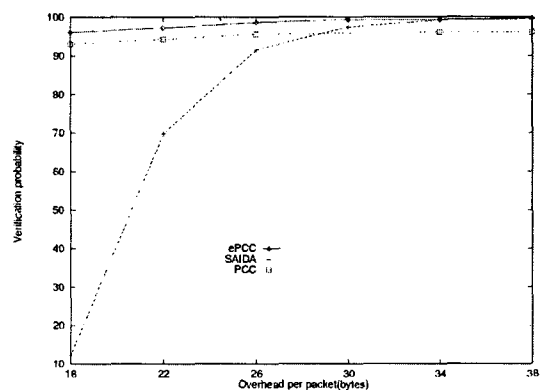


그림 3: 패킷 당 인증 정보량의 변화에 따른 검증확률 결과

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.

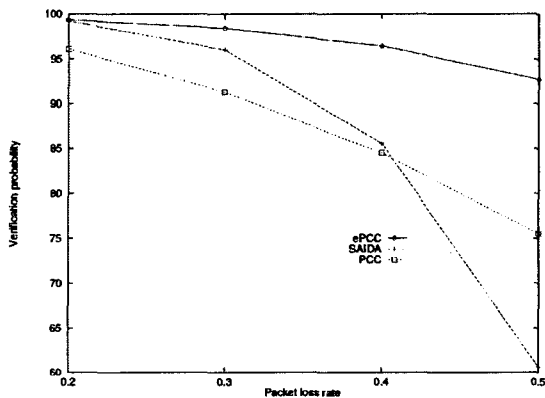


그림 4: 패킷 손실을 증가에 따른 검증확률 결과

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

## 2) 계산량 분석

본 절에서는 각 기법의 송신 서버와 수신자 수행 속도를 비교 분석한다. 세 기법 모두 해쉬 연산과 전자 서명 연산을 포함하며, 정보 분산 및 복원 연산은 SAIDA와 제안 기법에만 포함된다. 표 1은 각 기법을 구현한 후 단일 그룹에 대한 수행 시간을 측정된 결과이다. [4]에 따라 그룹의 크기  $n=128$ 로 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<sup>++</sup> 4.2, gcc version 2.96이다. 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 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 1: 송신 서버 및 수신자의 수행 속도 비교

패킷당 오버헤드 (바이트)	기법	송신 서버 수행시간 (ms)	수신자 수행시간 (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	1.81
	제안 기법	53.88	3.04
34	SAIDA	77.68	2.95
	PCC	23.72	1.78
	제안 기법	53.11	2.63
38	SAIDA	80.33	2.85
	PCC	23.93	1.86
	제안 기법	49.48	2.98

## III. 결론

본 논문에서는 최근에 발표된 스트림 인증 기법 중 하나인 PCC 기법을 정보 분산 알고리즘을 이용하여 개선한 방법을 제안했다. 제안된 기법은 PCC 기법의 서명 패킷에 정보 분산 알고리즘을 적용함으로써 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.

## 참고문헌

- [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] A. Perrig, R. Canetti, D. Song, and D. Tygar. Efficient authentication and signature of multicast streams over lossy channels. In proceedings of the IEEE Symposium on Research in Security and Privacy, May 2000.
- [3] 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.
- [4] 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.
- [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.