효율적인 플로우 및 멀티캐스팅 인증 기법 방안

백재종○ 강명호 송주석
연세대학교 컴퓨터과학과
{jjb27, jupiter, jssong}@emerald.yonsei.ac.kr

Efficient Digital Signature Scheme for Flows and Multicasts

J.J. Baek○ M.H. Kang J.S. Song
Dept. of Computer Science, Yonsei University

요 약

처리시간 지연에 민감한 패킷 플로우, 비디오, 오디오 스트림과 같은 어플리케이션을 보안성이 없는 인터넷 상에서 멀티 캐스팅 할 때에 데이터의 기밀성, 메시지 또는 발신자 인증, 무결성 그리고 부인 방지 등의 보안 서비스를 제공하기 위해서는 여러 가지 암호 및 인증 기법이 필요하다. 본 논문에서는 이러한 멀티캐스트 스트림과 플로우에 대한 서명/인증 기법의 특성과 요구사항을 분석해보고 기존에 제시된 each-sign 기법, one-time 서명 기법, star-chaining 기법, 그리고 Tree-chaining 기법에 대한 상호 비교 및 분석을 통해 이들의 서명/검증 계산시간과 통신 트래픽 오버헤드에 대한 단점을 효율적으로 개선하여 Enhanced Tree-chaining 기법을 제안한다. 서명 및 검증 시간은 약 50% 단축되며 통신 오버헤드는 $\log_2 n$ 배 축소되는 향상을 기대 할 수 있다. 또한 인증 소요 시간에 주요 요인이 되는 것은 서명/검증의 계산시간이 아니라 chaining 오버헤드의 크기임을 유추해 낸다.

1. 서 론

디지털 데이터 스트림은 메시지의 길이가 잠재적으로 매우 긴 비트열로써 송수신자간의 협상에 의해 전송속도를 결정하거나 상호 통신 프로토콜을 가지고 이에 따라 주기적으로 통신이 이루어진다. 일반적으로 수신측은 작은 버퍼를 가지고 있어서 수신한 스트림에 대한 압축 및 인증 시에 필요한 시간을 확보하게 된다.

일반적인 스트림에 대한 서명기법으로는 먼저 스트림을 일정한 크기의 블록으로 분할한 후 송신자는 각 블록을 서명하고 수신측에 전송한다. 수신자는 버퍼 크기만큼의 블록을 적재하여 먼저 각 블록에 대한 서명을 검증한 후 스트림을 사용하게 된다.

이러한 단순한 방법은 스트림이의 길이가 한정적이지 않을 때에도 적용이 가능하지만 계산적으로 비용이 매우 크며 버퍼의 크기에 따른 지연 시간 등 서명과 검증하는 연산은 스트림의 전송 속도에 비해 병목현상을 초래 할 것이다. 송신측에서 길이를 알 수 있는 유한 길이 스트림에는 다음과 같은 인증 시나리오를 생각해 볼 수 있다.

먼저 스트림을 블록으로 쪼갠 후 각 블록을 서명하는 대신에, 송신자는 각 블록에 대한 암호화된 해쉬값을 나열한 테이블을 생성하여 이 테이블에 서명을 한다. 그 다음 인증된 스트림을 요청하는 수신자에게 송신자는 이 테이블을 먼저 보낸다. 수신자는 이 테이블을 저장하고 테이블에 대한 서명을 검증한다. 따라서 전송된 각 블록은 수신지에 도착하자마자 검증될 수 있다. 하지만 이 방식의 문제는 메모리와 매우 큰 테이블에 대한 관리이다. 가장 먼저 전송되는 해쉬 테이블이 클 경우 치명적인 지연 시간을 초래 할 것이다.

2. One-time 서명 기법[2]

앞에서 살펴본 바 와 같이 일반적인 서명기법을 적용할 경우 해결해야 될 여러 가지 문제점을 가지고 있다. 먼저 유한 또는 무한 길이의 스트림에서 적용되어져야 하며, 큰 테이블을 먼저 전송함으로 저장공간이나 관리 메커니즘을 요구하지 않아야 한다. 또한 스트림의 길이가 인증에 사용되는 정보의 크기와는 관련이 없어야 한다. One-time 서명 기법은 이러한 요구사항을 만족하는데 i^{th}블록은 $(i+1)^{th}$블록을 인증하는 인증 정보를 가지게 하고 패킷의 시퀀스에 대해 단 한번의 고비용 서명/검증 연산을 하며 각 패킷에 대해서는 저비용의 One-time 서명/검증을 한다. 하지만 One-time 서명 기법은 많은 통신 오버헤드를 포함하는데 약 1000bytes/packet 이 되는 단점이 있다.

3. 멀티 캐스트 및 플로우 인증의 특성 과 요구사항.

현재의 TCP/IP 기반의 Best-effort 환경에서는 인터넷의 각 서비스에 따른 IP 패킷의 구분과 스케쥴링, 대역폭 보장등 QoS 기능이 없어 모든 패킷을 동일하게 처리된다. 이러한 환경에서의 플로우 및 멀티 캐스트 인증 기법의 특성들은 다음과 같다.

1) 플로우에서의 각 패킷은 수신과 동시에 처리되어야 한다.
2) 수신자는 자신의 패킷 서브 시퀀스만을 받아야 한다.
3) 처리시간 지연에 민감한 플로우는 수신측에서 신속한 처리를 요구하며 실시간에 생성되는 플로우는 송신자 역시 신속한 처리를 요구한다.

4) 멀티캐스트 플로우에 많은 수신자들은 송신자에 비해 시스템의 자원, 처리 능력, 메모리, 통신 대역폭 등에 한계성을 가지고 있다.
5) 수신자의 처리능력 및 자원에 따라 매우 다양하게 분포된다.

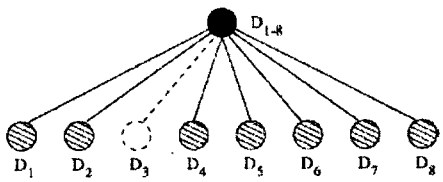
이와 같은 주어진 특성에 따라 서명/검증 스키마는 설계되어야 하며 다음과 같은 요구사항을 만족시켜야 한다.

1) 서명 절차는 효율적이며 실시간 생성된 플로우에 대해 지연이 제한(delay-bounded)되어야 한다.
2) 검증 절차는 수신자의 제한된 자원을 고려하여 효율적이어야 한다.
3) 플로우 내의 각 패킷은 각각 검증될 수 있어야 한다.
4) 패킷의 서명크기는 작아야 한다. (통신 오버헤드를 줄이기 위해)
5) 보안 등급은 조절이 가능하며 증가시킬 수 있어야 한다. 즉 검증은 수신자 자원의 양에 따라 적합하게 조절되어야 된다.

3. Chaining 스키마[1]

위에서 살펴본 멀티 캐스트 및 플로우 인증의 특성 과 요구사항을 기반으로 Star-Chaining 및 Tree-Chaining 스키마가 제안되었다. Chaining 기법은 각 패킷에 Chaining 정보를 유지하여 선행 인증된 노드에 의해 캐쉬된 해쉬값을 이용하여 해쉬 계산값을 줄이는 것이 기본 아이디어이다.

3.1 Star-Chaining



[그림 1] Star-chaining 기법

한 블록을 구성하는 패킷 다이제스트는 각 패킷에 대한 해쉬값이 되며 블록 signature는 각 패킷의 해쉬된 값을 concatenation하여 다이제스트 취한 값에 서명을 한 것이 된다. 패킷 다이제스트와 블록 다이제스트 사이의 관계는 authentication star라는 root 노드에 의해 표현된다. [그림 1] 는 8개의 leaf 노드 인 패킷 다이제스트로 구성된 authentication star를 보여준다. 각 패킷이 검증 되도록 하려면 각 패킷은 자신의 인증 정보가 필요하며 이를 패킷 signature 라고 부르고, 특히 모든 패킷 signature를 chaining 오버헤드라고 부른다. 이는 블록 signature, 블록에서의 패킷 위치, 그리고 블록에서 모든 패킷의 다이제스트로 구성된다. Star-Chaining 기법의 경우 Chaining 시간이 $O(m^2)$이 된다.

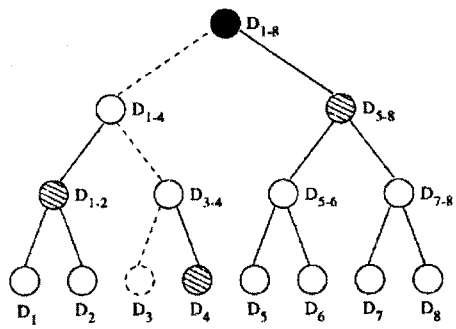
3.2 Tree Chaining

Tree chaining 기법은 원본 메시지에 자신의 인증 정보(Authentication info)를 포함시켜 전송하는 방식이다. 이러한 인증정보를 패킷 "signature"라고 부르며 내부에는 tree를 구성할 때 필요한 chaining 오버헤드를 부분을 포함하고 있다. 따라서 각 패킷은 개별적으로 수신지에서 검증되기 위해 인증정보에 포함하고 있는 다른 패킷들에 대한 해쉬값을 계산 및 비교하여 자신의 identity를 인증하게 된다. 하지만 긴 플로우를 전송할 때 이러한 chaining 오버헤드의 크기는 network 의 트래픽을 가중시키며 송수신 지에서의 서명 및 검증 시간을 소비하게 된다. 따라서 블록에 포함되는 chaining의 오버헤드를 감소시켜서 실시간 플로우 및 멀티 캐스트 전송의 효율을 높일 수 있다.

Tree-Chaining에서의 인증 정보는 블록 signature, 패킷의 위치, root로부터 연결된 패킷의 경로에서 각 노드의 형제노드(sibling)의 해쉬값들을 포함한다. 그 구조는 다음과 같다.

|<-------- chaining 오버헤드------ >|

다음 [그림 2]는 차수가 2, 깊이가 4인 이진 tree를 구성하여 서명/검증할 때, 각 패킷이 갖는 chaining 오버헤드는 다음과 같다. 전체 노드 수는 $2^n - 1$이며 n은 깊이(depth) 이다.



[그림 2] Tree-chaining 기법

각 패킷이 가져가는 chaining 오버헤드는 다음과 같다.

D_1: D_2, D_{34}, D_{58}, D_{18}, $Sign(D_{18})$
D_2: D_1, D_{34}, D_{58}, D_{18}, $Sign(D_{18})$
D_3: D_4, D_{12}, D_{58}, D_{18}, $Sign(D_{18})$
D_4: D_3, D_{12}, D_{58}, D_{18}, $Sign(D_{18})$
D_5: D_6, D_{78}, D_{14}, D_{18}, $Sign(D_{18})$
D_6: D_5, D_{78}, D_{14}, D_{18}, $Sign(D_{18})$
D_7: D_8, D_{56}, D_{14}, D_{18}, $Sign(D_{18})$
D_8: D_7, D_{56}, D_{14}, D_{18}, $Sign(D_{18})$

위 스키마를 예로 들면 다음과 같다. 세 번째 패킷이 도착하면 이의 D_3를 구하고 오버헤드에 포함해온 D_4 값과 다시 해쉬값을 구한다. 이 값은 다시 형제노드 D_{12}값과 해쉬를 취하여 D_{14}값을 구하고 D_{58}과 해쉬값을 구해 최종적으로 D_{18}과 비교하여 인증을 하게 된다. 이와 같이, 각 패킷이 갖는 chaining 오버헤드는 서로 중복되어 있으며, 실제로 사용되지 않으면서 포함되어 전송된다. 하지만 최초 패킷의 단 한번 서명/검증됨으로 다음에 수신되는 패킷들의 오버헤드는 불필요하게 된다. 이러한 비효율성을 제거하기 위해 다음과 같은 Enhanced Tree-Chaining 스키마을 제안한다.

4. Enhanced Tree-Chaining 스키마

제안하는 스키마는 기존의 Tree-Chaining 스키마 보다 더 적은 오버헤드를 가지고 전송되도록 하는 것이 기본 접근 방법이다. 즉 기존의 Tree-Chaining 스키마는 각 패킷이 자신과 조상들의 형제노드를 가지고 중복 전송되어 오버헤드로 인한 통신 트래픽 낭비의 단점이 있었다. 이를 착안하여 각 패킷은 선행 패킷이 가지고 가는 오버헤드와 비교하여 자신에게 필요한 해쉬값만을 가지고 전송하도록 한다. 검증은 먼저 기본적으로 root 해쉬값과 블록 signature 값을 비교하여 무결성을 인증한 다음, 모든 패킷이 수신되어 생성된 순서대로 검증하게 된다. 즉 각 패킷의 순서가 정렬되기 전까지 각 패킷은 무결성에 대한 인증을 미리 해놓은 다음, 블록에 속한 모든 패킷이 도착했을 때 일괄적으로 실행되는 메커니즘이다. 이는 해쉬시간이 고속으로 진행되며 입력되는 오버헤드의 사이즈를 출임으로 처리시간을 향상될 수 있다는 근거에 중점을 두고 있다. 따라서 제안하는 스키마에서의 chaining 오버헤드는 다음과 같다.

D_1: D_2, D_{34}, D_{58}, D_{18}, $Sign(D_{18})$
D_2: D_{18}, $Sign(D_{18})$
D_3: D_4, D_{18}, $Sign(D_{18})$
D_4: D_{18}, $Sign(D_{18})$
D_5: D_6, D_{78}, D_{18}, $Sign(D_{18})$
D_6: D_{18}, $Sign(D_{18})$
D_7: D_8, D_{18}, $Sign(D_{18})$
D_8: D_{18}, $Sign(D_{18})$

이와 같이 각 패킷에 포함되는 인증정보에는 기존 기법보다 적은 양이 필요하므로 훨씬 효율적이다. 기존 기법의 오버헤드 크기는 2^{d-1}개의(d:depth) leaf 노드를 가지므로 이진 tree의 오버헤드 크기는 $(\log_2 n+2)\times n$이 되어 $n\log_2 n+2n$(n: leaf node 개수) 된다. 반면에 새로 제안하는 Enhanced Tree-Chaining 의 오버헤드는 $(\log_2 n+2)\times \log_2 n$이 되어 $O(\log_2 n)^2$이 됨으로 기존의 크기보다 $\log_2 n$ 배 줄어듦을 알 수 있다. 각 패킷에 대한 오버헤드를 추출하는 알고리즘 및 프로시저는 다음과 같다.

ALGORITHM : tree-chaining(n, list(1,n))
Let n be the number of packets in a block.
Let H[i~i] = H[i].
Let i be the index of the leaf node.
For each leaf node i,

step 1) if (i-1 mod 2^1) = 0, then attach the hash value;
H[i+ 2^0 ~ i-1+ 2^1]
step 2) if (i-1 mod 2^2) = 0, then attach the hash value;
H[i+ 2^1 ~ i-1+ 2^2]
step 3) if (i-1 mod 2^3) = 0, then attach the hash value;
H[i+ 2^2 ~ i-1+ 2^3]
.....
step log n) if (i-1 mod 2^(log n)) = 0, then attach the hash value;
H[i+ 2^((log n)-1) ~ i-1+ 2^(log n)]

PROCEDURE : tree-chaining(n, list(1,n))

```
for k = 1 to (log n)
   for i = 1 to n
      if (i-1 mod 2^k) = 0, then attach the hash value;
                  H[i+ 2^(k-1) ~ i-1+ 2^k]
   end for.
end for.
```

5. 해쉬계산에 사용되는 입력크기 비교

다음은 [그림 2]에서 각 leaf 노드에서의 해쉬 회수를 통해 입력되는 크기를 비교 해보겠다. 첫 번째 패킷 P_1이 처음 도착할 때를 가정하면, 4회의 해쉬계산과 1회의 서명을 통해 비교하여 검증하게 된다. 이와 같이 수신지에서의 연속적인 패킷 수신에 따른 각 해쉬계산 회수 및 서명 회수는 다음과 같다.

P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8
4h, 1s	1h	2h	1h	3h	1h	2h	1h

(※ h 은 1회 hash time, s는 1회 signing time)

해쉬 함수로는 MD5를 사용하고 서명/검증 알고리즘으로 512 bit RSA를 사용한다고 가정하자. 이들의 결과 값의 크기는 각각 16byte, 64byte 가 된다.
Hash size H = 16 byte

Signature size S = 64 byte ∴ S= 4H 가 된다.
기존의 tree chaining 스키마에서의 입력 크기에 따른 계산회수를 구해보면 다음과 같다.
$n\times((1+\log_2 n)\cdot H+S)$, S=4H (n 은 블록수)
$\cong(5n+n\log_2 n)H$
if block size=64 ⇒ ≅(11n)H
if block size=128 ⇒ ≅(12n)H

새로 제안된 Enhanced tree chaining 기법에서의 해쉬계산의 입력 크기는 다음과 같다.
H(n-1) + Hn +Sn = H(2n-1) + Sn
= (2n-1)H + nS, S=4H ∴ (6n-1)H

이와 같은 결과의 식에서 알 수 있는 것은 새로 제안된 기법이 기존의 해쉬 해야할 오버헤드 크기를 약 50% 이상 단축 시켰음을 알 수 있다.
각 노드의 해쉬 계산은 공개키 알고리즘으로 서명하는 시간에 비하면, 상당히 작은 것이다. 또한, leaf 노드들을 제외한 internal 노드들의 해쉬 계산시 입력크기는 MD5를 사용할 경우 16-byte, SHA-1을 사용할 경우 20-byte로 일정한 크기가 된다. 이는 1회 해쉬 한 것과 3회 해쉬하는데 걸리는 시간은 무시 할 수 있는 아주 작은 시간의 차이를 의미한다. 따라서 서명/검증 시간에 주요 요인이 되는 것은 계산 시간이 아니라 chaining 오버헤드의 크기에 의해 크게 좌우됨을 알 수 있다.

6. 결 론

본 논문은 플로우 및 멀티캐스팅으로 전송되는 데이터 스트림에 대한 효율적인 서명/인증 스키마를 제안하였다. 이는 비 순서적으로 도착하는 패킷에 대한 개별적인 인증을 제공하면서 패킷에 포함되는 통신 및 계산 오버헤드의 크기를 줄임으로 서명/인증시간을 단축시켰다. 기존의 tree chaining 기법을 개선하여 해쉬 해야할 오버헤드들의 크기를 반 이상 단축시켰으며 통신 오버헤드를 약 $\log_2 n$ 배 축소 시켰다.
송수신에서 지연 없이 실시간의 통신을 지원하면서 데이터에 대한 보안 서비스를 보장하기 위해서는 더욱더 빠른 서명/검증 스키마를 구성하는 암호알고리즘 개발이 필요하며, 불필요한 통신 및 계산 오버헤드를 줄이는 방식으로 접근 해 나갈 수 있을 것이다.

7. 참고 문헌

[1]R.Gennaro and P.Rohatgi, "How to sign digital streams" in Advances in Cryptology-Crypto'97,1997,pp180-197
[2]R.C Merkle,"A Certified digital signature",in Advances in Cryptology-CRYPRO'89,1989,pp.218-238
[3]S.Mittra and T.Y.C woo,"A flow-based approach to datagram security" in Proc.ACM SIGCOMM'97,Cannes, France,1997,pp 221-234
[4]S.Even,O.Goldreich,S.Micali,"On-Line/Off-Line Digital Signatures",J.of Cryptology,9(1):35-67,1996
[5]C.K. Wong and S.S. Lam, "Digital Signature for Flows and Multicasts", IEEE/ACM TRANSACTION ON NETWORKING, VOL.7, NO.4, Aug 1999
[6]A.Perrig,R.Cannetti,J.D.Tygar,Dawn Song,"Efficient Authentication and signing of Mulicast streams over Lossy Channerls",2000 IEEE