비대칭 트리 구조를 이용한 멀티캐스트 암호화 키 관리

박희안[0], 공은배
충남대학교 컴퓨터공학과
{hapark,keb}@ce.cnu.ac.kr

Key Management for Multicast Encryption using Unbalanced Tree

Hee-An Park[0], Eun-Bae Kong
Dept. of Computer Engineering, Chungnam National University

요 약

본 논문에서는 안전한 멀티캐스트를 위해 사용되는 암호화 키를 비대칭 Tree 를 이용하여 관리하는 방법을 제안하였다. 기존의 대칭 Tree 를 이용한 방법은 모든 사용자들을 같은 레벨에 위치시킴으로써 각 사용자들에 따라 다를 수 있는 Join/Remove 빈도수를 고려하지 않고 모두 똑 같은 확률을 가진 것으로 간주하여 그에 따른 불필요한 메시지 전송이 많았다. 그러나 본 논문에서 제안한 방법은 각 사용자의 빈도수를 고려하여 Tree 를 비대칭으로 구성함으로써 키 관리를 위해 전송하는 메시지의 양을 효율적으로 줄일 수 있고 경우에 따라서는 Center 와 사용자가 가진 키의 양도 기존 방법보다 상당히 작아질 수 있다.

1. 서론

1 대 1 통신만 가능한 Unicast 방식보다 여러 사용자들에게 동시에 정보를 전달할 수 있는 Broadcast 방식과 Multicast 방식은 인터넷에서 가능한 여러 가지 서비스를 적용하기에 적합한 방법인데 그 중 Multicast 방식은 특정 Group 에게만 서비스가 가능하다는 장점 때문에 많은 서비스들에 적용되고 있는 방식이다. 이러한 Multicast 를 안전하게 사용할 수 있도록 하는 *Multicast Encryption* 이라는 방법은 *Group* 밖의 사용자로부터 정보를 보호하고 Group 내에 있는 사용자들만이 안전하게 정보를 공유할 수 있도록 하여 인터넷 방송과 같은 많은 유료 서비스에 적용할 수 있는 유용한 방법이다. Multicast Encryption 을 사용하는데 있어서 가장 중요한 사항은 정보의 암호화를 위해 사용되는 키를 어떻게 안전하고 효율적으로 관리하는가 하는 것인데 이를 위해 현재 많은 연구들이 진행되고 있고 그 중 가장 효율적인 방법이 Tree 를 이용한 방법이다. 즉, Tree 의 leaf 를 각각의 사용자와 연결하고 Group 을 관리하는 Center 를 Tree 의 Root 로 간주하여 전체 Group 을 하나의 Tree 로 구성하는 것이다. 이렇게 함으로써 사용자와 Center 가 가져야 하는 키의 양과 키를 변경할 때 보내지는 메시지의 전송량 사이의 tradeoff 를 적절히 조절하여 효율적인 Multicast Encryption 을 가능하게 한다.[1][2][3]
그러나 이 방식들은 모든 사용자들이 같은 확률을 가지고 Session 에 참여 한다고 간주하여 키 변경 시 발생하는 메시지 전송을 비 효율적으로 한다는 단점이 있다. 이러한 단점을 제거하기 위해 본 논문에서는 [3]에서 사용한 방식을 변형 시켜 Tree 를 비대칭으로 구성하고 각 사용자에게 Weight 를 주어 서로 다른 레벨의 leaf 를 할당함으로써 키 변경 시 필요한 메시지의 전송량을 효율적으로 줄일 수 있는 방법을 제안한다.
본 논문은 다음과 같이 구성되어 있다. 2 장에서는 Multicast Encryption 과 관련된 현재의 연구들을 간략히 살펴보고 3 장에서는 본 논문에서 제안하는 방법에 대해 살펴볼 것이며 4 장에서는 기존의 방법과 본 논문에서 제안한 방법을 실험을 통해 비교해볼 것이다.

* 이 연구는 충남대학교 정보통신인력양성사업단의 RA 지원금에 의해 수행되었음

2. 관련연구

이번 장에서는 키 관리를 위한 대표적인 두 가지 방법을 살펴보고, 이 두 가지 방법을 결합한 Combined a-ary Tradeoff Scheme[3]을 설명한다.

2.1 Minimal Storage Scheme

각각의 사용자는 Center 와 자신만이 공유하는 symmetric 키와 전체 사용자가 공유하는 session 키를 갖고, Root 는 session 키로 사용되는 자신의 키와 각 사용자들의 키를 생성하기위해 seed 값으로 사용되는 secret 키를 가진다. Group 내에서 Center 가 session 키를 이용하여 데이터를 암호화한 후 모든 사용자에게 Multicast 하면 각각의 사용자는 자신이 가지고 있는 session 키로 데이터를 복호화하는 방식으로 통신이 이루어진다. 만약 새로운 사용자가 Group 내에 Join 한다든지 기존의 사용자가 Group 에서 Remove 할 경우 Group 전체에서 사용하던 session 키를 바꾸어야 한다. 이런 경우 Root 는 session 키를 변경하고 각 사용자가 가지고 있는 symmetric 키로 새로운 session 키에 대한 정보를 암호화하여 Group 내에 남아있는 모든 사용자에게 각각 전달하여 Group 전체에서 사용되는 session 키를 변경한다. Minimal Storage Scheme 은 *Root* 와 사용자가 가지고 있는 키의 수가 적다는 장점이 있지만 키가 변경될 경우 새로운 키를 전송해야 하는 메시지 전송량이 매우 크다는 단점이 있다.

2.2 Basic Tree Scheme

Basic Tree Scheme 은 [1],[2]에서 소개하는 일반적인 Tree 방식으로 Center 를 Root 로, 각 leaf 를 사용자로 하여 전체 Group 을 Tree 로 구성하는 방식을 말한다. 각 사용자는 Center 와 공유하는 symmetric 키, Group 전체의 session 키, 그리고 자신에게 할당된 leaf 로부터 Root 까지의 path 에 있는 모든 노드들에게 할당된 키를 갖게 되고 Root 는 Tree 를 구성하는 전체 노드들에게 할당된 모든 키를 갖는다. 통신방법은 Minimal Storage Scheme 과 동일하고 만약 새로운 사용자가 *Group* 내에 *Join* 한다든지 기존의 사용자가 *Group* 에서

Remove 할 경우 Root 는 해당 사용자에게 새로운 leaf 를 할당하거나(Join 시) 사용자가 가지고있던 leaf 를 Tree 에서 제거하고(Remove 시) 그 사용자가 가져야 할 노드들의 키, 혹은 그 사용자에게 할당했던 노드들의 키를 변경한다. 그런 후 변경된 노드들의 자식 노드들의 키로 새로운 메시지를 암호화하여 Group 내에 있는 전체 사용자에게 알린다. Basic Tree Scheme 은 새로운 키를 전송하는 전송량이 효율적으로 줄어든다는 장점이 있지만 Root 가 가지고 있어야 하는 키의 양이 너무 커진다는 단점이 있다.

2.3 Combined a-ary Tradeoff Scheme

Combined a-ary Tradeoff Scheme 은 Minimal Storage Scheme 과 Basic Tree Scheme 을 결합하여 보관해야 하는 키의 양과 키 변경 메시지 전송량 사이의 Tradeoff 를 적절하게 조절한 효율적인 키 관리 방식이다. 우선 Group 내의 모든 사용자를 m 명의 subgroup 으로 분리하고 각 subgroup 을 Tree 의 각 leaf 에 연결한다. 그림 1 은 이 방식을 도식화 한 것이다.

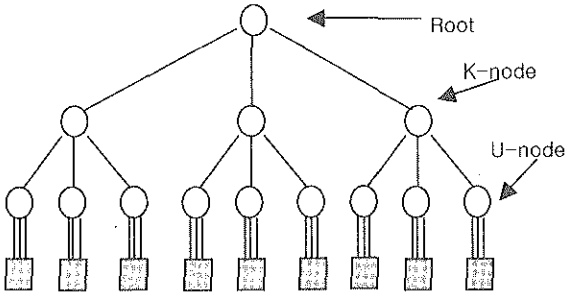

그림 1. Combined a-ary Tradeoff Scheme

그림 1 에서 검게 표시된 부분은 m 명으로 구성된 subgroup 을 나타낸다. Basic Tree Scheme 과 마찬가지로 각 사용자는 자신의 symmetric 키와 Group 의 session 키, 그리고 자신에게 할당된 leaf 로부터 Root 까지의 path 에 있는 모든 노드들에게 할당된 키를 갖게 된다. Root 는 전체 노드들의 키와 Group 의 session 키, 그리고 subgroup 내의 각 사용자들과 공유하는 symmetric 키를 생성하는데 seed 값으로 사용되는 secret 키를 U 노드만큼 갖게 된다. 통신방법은 Basic Tree Scheme 과 동일하고 만약 Group 내에서 Join/Remove 가 발생하면 Root 는 해당 U 노드의 키를 변경한 후 그 U 노드에 매달린 남아있는 사용자에게는 Minimal Storage Scheme 방식으로 다른 U 노드에 매달린 사용자에게는 Basic Tree Scheme 방식으로 변경된 키를 전달한다. Combined a-ary Tradeoff Scheme 은 Basic Tree Scheme 방식보다 Tree 의 크기가 훨씬 작아지기 때문에 Root 가 가지는 키의 양이 적어지고 키 변경 메시지의 전송량도 Minimal Storage Scheme 방식보다 현저히 줄어드는 장점을 가지고 있다. 그러나 모든 사용자가 자신의 Join/Remove 빈도수와 상관없이 같은 레벨에 위치하기 때문에 Join/Remove 시 항상 같은 키 변경 메시지 전송량(Communication)이 필요하다는 단점이 있다. 이러한 단점을 보완하기 위해 본 논문에서는 Tree 를 비대칭으로 구성하고 각 사용자의 Join/Remove 빈도수를 고려하여 서로 다른 레벨에 사용자를 위치시킴으로써 *Combined a-ary* Tradeoff Scheme 보다 Communication 을 보다 더 줄일 수 있는 방법을 제안한다.

3. Unbalanced Tradeoff Scheme

이번 장에서는 본 논문에서 제안하는 Unbalanced Tradeoff *Scheme* 의 전반적인 개념과(3.1) *Combined a-ary Tradeoff* Scheme, Unbalanced Tradeoff Scheme 에 사용된 parameter 들을 이용하여 User storage, Center storage, 마지막으로 Communication 사이의 Tradeoff 를 비교한다(3.2).

3.1 Unbalanced Tradeoff Scheme

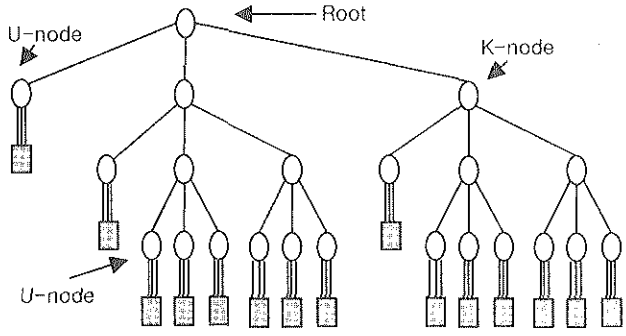

그림 2. Basic Unbalanced Tradeoff Scheme

그림 2 는 기본 Scheme 인 Basic Unbalanced Tradeoff Scheme 을 도식화 한 것이다. Basic Unbalanced Tradeoff Scheme 은 각 노드의 자식 중 하나를 U 노드로 놓고 나머지 자식은 K 노드로 놓는 식으로 Unbalanced Tree 를 끝까지 확장한 후 마지막 레벨에서는 모든 노드를 U 노드로 놓는 방식이다. 각 사용자는 자신이 Tree 에 Join/Remove 하는 빈도수에 따라 Weight 가 정해지고 Weight 가 큰(빈도수가 많은) 사용자는 레벨이 낮은 노드에 Weight 가 작은(빈도수가 적은) 사용자는 레벨이 높은 노드에 위치하게 된다.
Basic Unbalanced Tradeoff Scheme 의 데이터 전달방법과 Join/Remove 시 키의 변경방법, 변경된 키를 전달하는 방법은 Combined a-ary Tradeoff Scheme 과 동일하다.
Group 내에서 Join/Remove 가 발생할 경우 높은 레벨의 사용자보다 낮은 레벨의 사용자일 확률이 크게 되므로 새로운 키의 Communication 은 Combined a-ary Tradeoff Scheme 보다 줄어들게 되고 많은 수의 사람들이 Join/Remove 를 반복한다면 그 차이는 상당히 커질 것이다. 또 Unbalanced Basic Tradeoff Scheme 에서 각 노드에 매달리는 U 노드의 개수를 k 개로 늘리면 General Unbalanced Tradeoff Scheme 으로 확장할 수 있는데 이럴 경우 레벨이 낮은 곳의 사용자가 늘어나 그곳에서 Join/Remove 가 일어날 확률이 더욱 커지므로 Communication 양은 더욱 줄어들 것이다.

3.2 두 Scheme 의 비교

이 장에서는 두 Scheme 에서 공통으로 사용되는 parameter 들을 이용하여 Combined a-ary Tradeoff Scheme 과 General Unbalanced Tradeoff Scheme 의 User storage, Center storage, Communication 을 비교하였다.

- *가정* : 두 *Scheme* 의 m, n, a 값은 같다.
- Parameter
  - n : Group 내의 전체 사용자
  - m : subgroup 에 포함된 사용자

- a : Tree 의 Degree
- d : Tree 의 Depth
- k : Unbalanced Tradeoff Scheme 에서 한 노드에 배달리는 U 노드의 개수(Basic Scheme 에서 k=1)

- Combined a-ary Tradeoff Scheme 의 Depth

$$a^{depth} \geq \frac{n}{m}$$

$$depth \geq \log_a(\frac{n}{m})$$

- Unbalanced Tradeoff Scheme 의 Depth

$$m(0+k+k(a-k)+k(a-k)^2+...+k(a-k)^{depth-2}+a(a-k)^{depth-1}) \geq n$$

$$depth \geq \log_{(a-k)}(\frac{n(a-k-1)+mk}{m(k+a^2-ak-a)})+1$$

- Combined a-ary Tradeoff Scheme 의 User storage
  - $depth + 2$

- Unbalanced Tradeoff Scheme 의 User storage
  - min : 3 max : $depth + 2$

- Combined a-ary Tradeoff Scheme 의 Center storage
  - U 노드의 seed 로 쓰이는 secret 키 + 전체 노드 수

$$\frac{n}{m} + (1 + a + a^2 + ... + a^{depth-1} + a^{depth}) = \frac{n}{m} + \frac{a^{depth+1}-1}{a-1}$$

- Unbalanced Tradeoff Scheme 의 Center storage
  - U 노드의 seed 로 쓰이는 secret 키 + 전체 노드 수

$$\frac{n}{m} + (1 + a + a(a-k) + a(a-k)^2 + ... + a(a-k)^{depth-2} + a(a-k)^{depth-1}) = \frac{n}{m} + \frac{a(a-k)^{depth}-k-1}{a-k-1}$$

- Combined a-ary Tradeoff Scheme 의 Communication
  - Minimal Storage Scheme + Basic Tree Scheme
  - $m - 1 + (a-1)depth$

- Unbalanced Tradeoff Scheme 의 Communication
  - Minimal Storage Scheme + Basic Tree Scheme
  - min : $m + a - 2$ max : $m - 1 + (a-1)depth$

수식에서 보듯이 두 Scheme 모두 정해진 n 이 있을 때 m 과 a, 그리고 k 를 어떻게 선택하느냐에 따라 효율이 좌우된다는 것을 알 수 있다.

4. 실험

이번 장에서는 각 parameter 에 실제 값을 넣어 두 Scheme 에 따라 각각 Tree 를 구성했을 때 User storage, Center storage, Communication 이 어떻게 차이가 있는지 실험을 통해 알아보았다. 표 1 은 두 Scheme 모두 n = 100000, m = 20, a = 10 의 값을 가졌을 경우이고, 표 2 는 두 Scheme 모두 n = 100000, m = 100 의 값을 갖고 Combined a-ary Tradeoff Scheme 에서는 a = 10 으로, Unbalanced Tradeoff Scheme 에서는 a = 8 로 정하여 각각 최적의 Tree 를 구성했을 경우의 차이점을 보여준다.

	User	Center	Communication
Combined	6	16111	55
Unbalanced(k=5)	min : 3 max : 7	12811	min : 28 max : 64

표. 1 일반적인 경우의 Tradeoff 비교

	User	Center	Communication
Combined	5	2111	126
Unbalanced(k=3)	min : 3 max : 6	2249	min : 106 max : 127

표. 2 최적화한 경우의 Tradeoff 의 비교

표 1 에서는 k=5 일 때의 General Unbalanced Tradeoff Scheme 이 Combined a-ary Tradeoff Scheme 보다 Center storage 측면에서도 효율적일 수 있다는 것을 보여주고 있다. Tree 가 최적화 되어있는 표 2 에서 Center storage 가 어느 정도의 차이를 보이고 있는데 만약 DES 를 사용하여 키의 크기를 7byte 로 했을 경우 그 차이는 1Kbyte 이하가 되므로 충분히 감수할 수 있는 오버헤드가 된다.
아래 표 3 은 표 2 의 조건에서 Join/Remove 수를 10000 으로 했을 경우에 두 Scheme 의 Communication 을 비교한 것이다. 이때 Unbalanced Tradeoff Scheme 의 각 레벨에서 일어날 수 있는 Join/Remove 확률을 레벨 1 은 40%, 레벨 2 는 35%, 레벨 3 은 20%, 그리고 레벨 4 는 5%의 Weight 를 주고 계산하였다. 실제에서는 레벨이 낮은 곳에서 Join/Remove 가 더욱 집중적으로 일어날 것이므로 실제 수치와는 차이가 있을 수 있고, 또한 m 값이 줄어들면 그 차이를 더욱 확연히 느낄 수 있게 된다 표 3 에서는 약 11%의 Communication 양을 줄이는 효과를 보았다.

	Combined	Unbalanced	차이
표 2	1260000	1123000	137000

표. 3 Communication 의 비교

5. 결론

본 논문에서는 Unbalanced Tree 를 이용하여 키를 관리함으로써 Balanced Tree 로 구성했을 때보다 Communication 에서 훨씬 좋은 효율을 갖는 새로운 방식을 제안하였다. 정확한 자료수집으로 각 사용자의 Weight 를 적절히 조절하고 최적의 Tree 를 구성하기위해 적절한 parameter 들을 사용한다면 오버헤드를 최소한으로 줄이면서 Communication 효율을 최대한 증가 시킬 수 있는 Unbalanced Tradeoff Scheme 을 구현 할 수 있을 것이다.

6. 참고문헌

[1] D.Wallner, E.Harder, and R.Agee. "Key Management for Multicast : Issues and Architectures" RFC 2627, June 1999
[2] Chung Kei Wong, Mohamed Gouda, and Simon S. Lam. "Secure Group Communications Using Key Graphs" IEEE/ACM Transactions on Networking 8 (1): 16-30(2000)
[3] Ran Canetti, Tal Malkin, and Kobbi Nissim. "Efficient Communication-Storage Tradeoffs for Multicast Encryption" Volume 1592, Issue , pp 0459-LNCS