제18회 한국정보처리학회 추계학술발표대회 논문집 제9권 제2호 (2002.11)

스테이트리스 리시버를 위한 효율적인 멀티캐스트 키관리

기주희, 김현정, 이동훈, 박창섭
고려대학교 정보보호대학원
e-mail : eye@cist.korea.ac.kr

Efficient Multicast Key Management for Stateless Receivers

Ju Hee Ki, Hyun Jueong Kim,
Dong Hoon Lee and Chang-Seop Park
Center for Information Security Technology, Korea university

요 약

이 논문에서는 동적이고 규모가 큰 그룹에 대해서 한명의 그룹 관리자가 존재하면서, 특별히 새로운 그룹키가 갱신될 때마다 새로운 정보를 받기 힘든 구성원, 즉 스테이트리스 리시버(stateless receiver)에게 적합한 방법을 제안한다. 이 방법은 구성원에게 각각 한 개씩 주어지는 개인키를 전송하는 메시지를 제외한 다른 모든 메시지들에는 암호화 과정이 요구되지 않는다. 즉, 갱신된 그룹키를 공유하기 위해 필요한 계산은 단지 $O(\log_2 n)$ 번의 해쉬함수 계산과 배타적 논리합(XOR)을 수행하는 것이며, 그룹키를 갱신하기 위해 필요한 정보는 암호화될 필요없는 멀티캐스트 메시지와 그룹에 추가될 때 그룹 관리자로부터 받은 초기값이다. 또한 제안하는 방법은 새롭게 추가된 사람이 이전의 그룹키에 대한 어떠한 정보도 알 수 없으며(후방보호 : Backward Secrecy), 삭제되는 사람 역시 이후의 새로운 그룹키에 대한 정보를 알 수 없다(전방보호 : Forward Secrecy). 또한 제안된 방법에 게시판이 이용된다면, 각 그룹의 구성원은 어떠한 멀티캐스트 메시지없이 단지 자신의 초기 개인키만으로 필요한 모든 노드키들을 계산할 수 있다.

1. 서론

멀티캐스트 통신 분야에서 가장 중요한 사항은 단지 합법적인 구성원만이 멀티캐스트 통신에 접근할 수 있어야 하기 때문에 공통적인 그룹키로 메시지를 암호화 한다. 그러나 그룹 구성원들이 실시간 추가되거나 삭제되기 때문에 그룹 통신을 위해 사용되는 암호키들는 그 때마다 갱신되어야 하는 어려움이 있다. 또한 그룹의 규모가 클수도 있기 때문에 멀티캐스트 통신 방법을 디자인할 때에는 그룹키를 위한 계산량이나 저장량은 효율적이어야 한다.

이 논문의 나머지는 3부분으로 나누어진다. 2장은 이 논문에서 제시하는 새로운 키관리 방법을 설명하고, 3장은 기존의 다른 방법과 비교를 함으로서 효율성을 알아보며, 4장의 결론을 통해 이 논문을 맺는다.

2 제안하는 방법의 기본 구조

이 논문에서 제안하는 기법은 기존의 다른 방법처럼 한 명의 그룹 관리자가 계층적 이진 트리를 관리한다. 트리 내부 노드 각각은 단지 두 개의 자식노드와 연결되어 있고, 말단노드는 그룹 구성원이 할당되며, 최상위 루트키는 그룹키로 정의한다. 이 논문에서 제안하는 기법에서는 아래와 같은 특성을 지닌 함수 $H: X \times Y \rightarrow Y$ 를 이용한다.

> ■ **특성 1** 모든 $x_1, x_2 \in X$, $y \in Y$ 에 대해 $H(x_1, H(x_2, y)) = H(x_2, H(x_1, y))$ 를 만족한다.
>
> ■ **특성 2** 모든 $x \in X$, $y \in Y$ 에 대해 y와 $z\ (= H(x, y))$ 가 주어졌을 때, $H(x', y) = z$ 를 만족하는 $x' \in X$ 를 찾기 어렵다.

예) $H(x, y) = h(x) \oplus y$ (단, h 는 일방향 함수이고,

⊕ 는 배타적 논리합이다.)

그룹을 형성하기 위해서 먼저 그룹 관리자는 각각의 그룹 구성원 U_i 마다 랜덤한 개인키 k_i 를 트리의 말단노드 n_i 에 할당한 후, 각 구성원들에게 유니캐스트로 보내준다. 또한 그룹 관리자는 k_i 와 관련된 보조키 r_i^*를 n(그룹 구성원의 수)개 랜덤하게 선택한다. 트리 내부의 노드키 k_j 를 생성하기 위해 그룹 관리자는 각 내부 노드마다 r_j 를 랜덤하게 선택한 후 k_j를 다음과 같은 규칙에 의해 계산한다.

n_j 가 말단 노드의 부모노드인 경우에는,

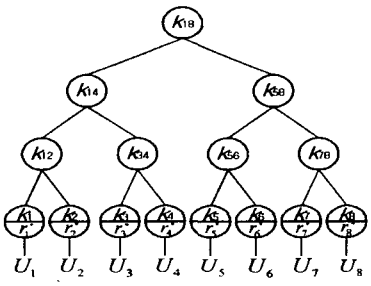
$$k_j = H(k_{L(j)} \oplus r^*_{L(j)}, H(k_{R(j)} \oplus r^*_{R(j)}, r_j)$$
$$= H(k_{R(j)} \oplus r^*_{R(j)}, H(k_{L(j)} \oplus r^*_{L(j)}, r_j)$$

그 이외의 경우에는,

$$k_j = H(k_{L(j)}, H(k_{R(j)}, r_j) = H(k_{R(j)}, H(k_{L(j)}, r_j)$$

편의상, $H(k_{L(j)}, r_j)$ 와 $H(k_{R(j)}, r_j)$ 를 k_j 의 블라인드 요소들(blinded factors)이라 정의하며, 그룹 구성원 U 의 말단노드부터 루트노드까지의 경로 중에 k_j 와 $k_{R(j)}$ 가 있다면, $H(k_{L(j)}, r_j)$ 를 $k_{R(j)}$의 블라인드 형제 요소(blinded sibling factor)라 정의한다.

그룹 관리자는 그룹 구성원 각각에게 안전하게 개인키를 유니캐스트한 후에, n개의 보조키들 $[r_1^*, \cdots, r_n^*]$와 루트키를 제외한 트리 내부의 노드키들과 관련된 $(n-1)$개의 블리인드 요소 $[H(k_{L(j)}, r_j), H(k_{R(j)}, r_j)]$들을 멀티캐스트 한다. 그러면 모든 구성원은 자신의 개인키와 블라인드 형제 요소를 이용해 말단노드에서부터 루트노드까지의 경로에 있는 모든 노드키들을 계산할 수 있다.



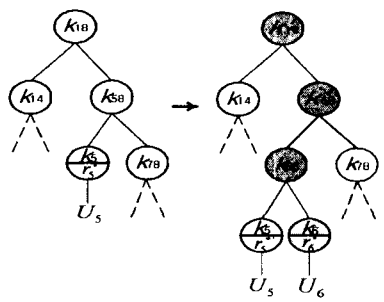
[그림 1] 그룹 초기화

예를 들어 [그림 1]에서 그룹 관리자는 구성원 U_i 에게 개인키 k_i 를 유니캐스트하고, 8개의 보조키와 7쌍의 블라인드 요소들을 멀티캐스트 한다. : $[r_1^*, r_2^*, r_3^*, r_4^*, r_5^*, r_6^*, r_7^*, r_8^*, H(k_1 \oplus r_1^*, r_{12}), H(k_2 \oplus r_1^*, r_{12}), H(k_3 \oplus r_3^*, r_{34}), H(k_4 \oplus r_4^*, r_{34}), H(k_5 \oplus r_5^*, r_{56}), H(k_6 \oplus r_6^*, r_{56}), H(k_7 \oplus r_7^*, r_{78}), H(k_8 \oplus r_8^*, r_{78}), H(k_{12}, r_{14}), H(k_{34}, r_{14}), H(k_{56}, r_{58}), H(k_{78}, r_{58}), H(k_{14}, r_{18}), H(k_{58}, r_{18})]$. $n_{L(j)}$ 을 기준으로 한 서브트리에 있는 모든 구성원들은 각자의 개인키를 이용해서 $k_{L(j)}$ 를 계산할 수 있다. 예를 들어, 개인키로 k_3 를 가지고 있는 구성원 U_3 은 다음과 같이 k_{34}, k_{14}, k_{18} 을 계산한다.:

$$k_{34} = H(k_3 \oplus r_3^*, H(k_4 \oplus r_4^*, r_{34}))$$
$$= h(k_3 \oplus r_3^*) \oplus h(k_4 \oplus r_4^*) \oplus r_{34}$$
$$k_{14} = H(k_{34}, H(k_{12}, r_{14})) = h(k_{34}) \oplus h(k_{12}) \oplus r_{14}$$
$$k_{18} = H(k_{14}, H(k_{58}, r_{18})) = h(k_{14}) \oplus h(k_{58}) \oplus r_{18}$$

2.1 구성원 추가

그룹 관리자는 새로운 구성원 U_i 의 가입 메시지를 받으면, U_i 의 말단노드로 할당할 n_i 를 트리에 삽입하기 위해 루트노드에 가장 가까이 있는 말단노드 n_j를 찾는다. 그리고 이 n_j 를 n_i 와 n_j의 부모노드 n_p 로 할당한다. 예로 [그림 2]에서 U_6 가 그룹에 추가될 때, 노드 n_5 가 루트노드와 가장 가까운 말단노드라고 하자. 새로운 노드 n_{56} 은 말단노드 n_5 와 n_6 의 부모노드가 되고, U_6 은 n_6 에 추가되며, n_6의 노드키는 그룹 관리자가 랜덤하게 선택한 k_6이 할당된다.



[그림 2] U6이 추가되는 경우

후방보호(Backward Secrecy)를 만족하기 위해 그룹 관리자는 n_{56} 에서부터 루트노드까지의 경로에

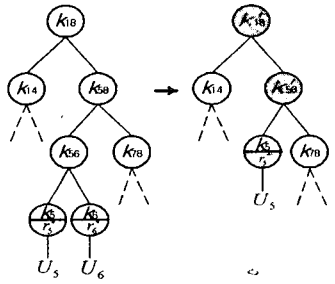
있는 모든 노드키들을 바꿔야 한다. 그룹 관리자는 U_6 의 보조키를 포함해서 다섯 개의 랜덤 수(r_6^*, $r_5^{*'}$, $r_{56}', r_{58}', r_{18}'$)를 선택한 후, 다음과 같이 계산한다.

$$k_{56} = H(k_5 \oplus r_5^{*'}, H(k_6 \oplus r_6^*, r_{56}))$$
$$= H(k_6 \oplus r_6^*, H(k_5 \oplus r_5^{*'}, r_{56}))$$
$$k_{58}' = H(k_{56}, H(k_{78}, r_{58}')) = H(k_{78}, H(k_{56}, r_{58}'))$$
$$k_{18}' = H(k_{14}, H(k_{58}', r_{18}')) = H(k_{78}, H(k_{56}, r_{18}'))$$

그룹 관리자는 U_6 에게 유니캐스트 메시지로 $[k_6]$ 을, 멀티캐스트 메시지로 $[r_5^{*'}, r_6^*, H(k_5 \oplus r_5^{*'}, r_{56}), H(k_6 \oplus r_6^*, r_{56}), H(k_{56}, r_{58}'), H(k_{78}, r_{58}'), H(k_{14}, r_{18}')), H(k_{58}', r_{18}')]$ 을 전송한다. 여러 명의 구성원이 추가되는 경우에도 이와 유사하다.

2.2 구성원 삭제

말단노드 n_i 에 할당되어 있던 그룹 구성원 U_i 가 삭제되는 경우, 전방보호(Forward Secrecy)를 만족하기 위해 n_i 의 부모노드부터 루트노드까지의 모든 경로에 있는 노드키는 갱신되어야 한다.



[그림 3] U6이 삭제되는 경우

[그림 3]에서 구성원 U_6 이 그룹으로부터 삭제되는 경우, U_6 의 말단노드 n_6 와 부모노드 n_{56} 은 제거되고 부모노드 n_{58} 에서부터 루트노드 n_{18}까지의 모든 노드키들은 갱신되어야 한다. 그룹 관리자는 U_6 의 형제인 U_5 의 보조키를 포함해서 세 개의 랜덤 수 ($r_5^{*'}, r_{58}', r_{18}'$) 를 선택한 후, 다음과 같이 노드키들을 갱신한다.

$$k_{58}' = H(k_5 \oplus r_5^{*'}, H(k_{78}, r_{58}'))$$
$$= H(k_{78}, H(k_5 \oplus r_5^{*'}, r_{58}'))$$
$$k_{18}' = H(k_{14}, H(k_{58}', r_{18}')) = H(k_{58}', H(k_{14}, r_{18}'))$$

그룹 관리자는 멀티캐스트 메시지로 $[r_5', H(k_{78}, r_{58}'), H(k_5 \oplus r_5', r_{58}'), H(k_{14}, r_{18}'), H(k_{58}', r_{18}')]$ 를 전송한다. 여러 명의 구성원이 삭제되는 경우도 이와 유사하다.

2.3 게시판을 이용하는 키관리 변형

만약 그룹 관리자가 모든 블라인드 요소들과 보조키들을 공고할 수 있는 공공 게시판을 이용한다면, 논문에서 제시하는 키관리 방법은 추가적인 장점을 지니게 된다. 게시판에 공고할 데이터의 양은 $(3n-1)K$ (단, K는 키길이) 이다. 이 때, 각 구성원은 단지 자신의 개인키만 소유하고 있으면 언제든지 그룹키를 계산할 수 있다.

또한 공공 게시판을 이용하면 구성원이 오프라인 상이거나 일정기간 그룹 관리자로부터 메시지를 받지 못하는 환경이더라도 언제든지 이 공공게시판의 데이터만 읽으면 현재의 그룹키를 얻을 수 있다. 게시판에 접근할 수 없는 구성원 U_i 에 대해서는 그룹 관리자가 단지 $\log_2 n$ 개의 블라인드 요소들과 현재의 보조키 r_i^* 만 보내주면 된다.

3 효율성 비교

트리 구조에 기반하는 키관리 방법은 트리의 높이에 크게 의존한다. 따라서 이 논문에서는 가능한 한 트리의 높이를 낮추기 위해 새롭게 추가되는 구성원들은 루트노드에 가장 가까운 말단노드에 추가시킨다. 다음은 이 논문에서 제시하는 방법과 Wallner[5] 와 Caronni[1,4]이 제안했던 HBT, McGrew와 Sherman 이 제안했던 OFT[2], Rafaeli가 제안했던 EHBT[3] 의 효율성을 비교한다. 트리는 모두 2진트리로 가정한다. 테이블 1, 2, 3, 4 는 그룹 관리자와 구성원 간의 멀티캐스트와 유티캐스트 양, 계산량을 각각 비교한다. 각 기법의 표기를 위해 다음과 같은 기호를 정의한다.

- n : 그룹 구성원의 수
- d : 트리의 높이
- K : 안전한 키의 길이

- I : 키 인덱스의 길이
- R : 키 생성 알고리즘 실행
- H : 일방향 함수 실행
- X : 배타적 논리합 실행
- E : 대칭 암호화에 의한 암·복호화 실행
- N_l : l명의 구성원이 추가되거나 삭제되었을 때, 갱신되어야 하는 노드키의 개수

논문에서 제시하는 키관리 방법은 그룹 관리자와 구성원들이 노드키의 갱신을 위한 메시지를 암호화하거나 복호화를 할 필요가 없다. 그룹키가 갱신될 때에 구성원에게 필요한 최대 계산량은 $\log_2 n$번의 해쉬함수의 계산과 배타적 논리합이다. 비록 이 방법이 기존의 OFT와 EHBT보다 멀티캐스트 양은 많지만, 계산량과 저장량, 스테이트리스 리시버의 측면에서 훨씬 효율적이다.

	Computation			Size of re-keying message	
	GC	Join member	Sibling	Join Unicast	Multicast
HBT	$(d+1)R+(2d+1)E$	$(d+1)E$	dE	K	$2dK$
OFT	$R+d(H+X)+(2d+1)E$	$(d+1)E+d(H+X)$	$E+d(H+X)$	$(d+1)K$	dK
EHBT	$R+(d+1)X+H+E$	$(d+1)E$	$(d+1)(X+H)$	$(d+1)K$	$4l$
Our Scheme	$R+2d(X+H)+E$	$E+2d(H+X)$	$d(H+X)$	K	$2dK$

[표 1] 한 명의 구성원이 추가되는 경우

	Computation			Size of re-keying message	
	GC	Join member	Sibling	Join Unicast	Multicast
HBT	N_lR+2N_lE-2E	$(d+1)E$	dE	$2K$	$2N_ldK$
OFT	$lR+(N_l-1)H+(N_l-1)X$ $+(N_l+1-1)E$	$(d+1)E+dH+dX$	$E+d(H+X)$	$Hd-1)K$	N_lK
EHBT	$lR+l(d-1)E+N_l(X+H)$	$(d+1)E$	$(d+1)(X+H)$	$H(d+1)K$	$N_l l$
Our Scheme	$lR-lE+2l(N_l-1)(H+X)$	$E+d(H+X)$	$dH+X)$	$2K$	$2N_lK$

[표 2] 다수의 구성원이 추가되는 경우

	Computation			Size of re-keying message	
	GC	Sibling	Max. other member (except sibling)	Sibling Unicast	Multicast
HBT	$dR+2dE$	dE	$(d-1)E$	K	$2dK$
OFT	$R+d(H+X+E)$	$E+d(H+X)$	$(d-1)(R+H+X)$	K	dK
EHBT	$d(X+H)+(d-1)K$	$d(X+H)$	$(d-1)K$	1	$4K$
Our Scheme	$2(d-1)(X+H)$	$d(X+H)$	$(d-1)(X+H)$	-	$2dK$

[표 3] 한 명의 구성원이 삭제되는 경우

	Computation			Size of re-keying message	
	GC	Sibling	Max. other member (except sibling)	Sibling Unicast	Multicast
HBT	$N_l(R+E)$	dE	$(d-1)E$	K	$2N_lK$
OFT	$lR+(N_l-1)(H+X+E)$	$dR+(d-1)(H+X)$	$(d-1)(R+H+X)$	K	N_lK
EHBT	$N_l(X+H)+(N_l-1)E$	$d(X+H)$	$(d-1)K$	1	N_lK
Our Scheme	$2(N_l-1)(X+H)$	$d(X+H)$	$(d-1)(X+H)$	-	$2N_lK$

[표 4] 다수의 구성원이 삭제되는 경우

4 결론

동적이고 규모가 큰 그룹을 안전하게 유지하기 위해, 이 논문에서는 새롭고 효율적인 알고리즘을 제안했다. 이 방법은 모든 계산 과정이 해쉬함수의 계산과 배타적 논리합으로 이루어지며, 그룹키를 포함해 노드키를 갱신하기 위한 멀티캐스트 메시지에 대한 암·복호화 과정이 없다. 그리고 기존의 다른 방법에 비해 스테이트리스 리시버에게 효율적이다. 더욱이 공공 게시판을 이용한다면, 적은 데이터의 공고만으로 멀티캐스트 메시지가 전혀 없이 그룹키 갱신이 가능하다. 또한 이 방법은 후방보호(Backward Secrecy)와 전방보호(Forward Secrecy) 둘다 만족한다.

참고문헌

[1] G. Caronni, M. Waldvogel, D. Sunand, and B. Plattner. Efficient Security for Large and Dynamic Multicast GRoups. In Workshop on Enabling Technologies, (WETICE 98). IEEE Comp Society Press, 1998.

[2] D. A. McGrew and A. T. Sherman. Key Establishment in Large Kynamic Groups Using One-Way Function Trees. Technical REport No. 0755, TIS Labs at Network Associates, Inc., Glenwood, MD, May 1998.

[3] S. Rafaeli, L. Mathy, and D. Hutchison. EHBT: An efficient protocol for group key Management. Proc. of 3rd Intl. COST264 Workshop on Networked Group Communication (NGC 2001) Lecture Notes in Computer Science 2233, Springer, pp. 159-171. 2001.

[4] M. Steiner, G. Tsudik, and M. Wakdner. Key Agreement in Dynamic Peer Groups. IEEE TRanscations on Parallel and Distributed System. March 2000.

[5] K. Wallner, E. Harder, and R. Agee. Key management for a multicast : Issues and Architectures RFC 2627, June 1999.