신뢰성 제공을 위한 그룹키 복구 메커니즘†

조태남*° 김상희* 이상호* 채기준* 박원주** 나재훈**
*이화여자대학교 컴퓨터학과, **ETRI 정보보호연구본부
{tncho, kshee, shlee, kjchae}@ewha.ac.kr, {wjpark, jhnah}@etri.re.kr

A Group Key Recovery Mechanism for Reliability

Taenam Cho*° Sanghee Kim* Sang-Ho Lee* Kijoon Chae* Wonjoo PARK** Jaehoon Nah**
*Dept. of Computer Science and Engineering, Ewha Womans University
**Information Security Technology Division, ETRI

요 약

그룹 통신 보안에 사용되는 그룹키의 갱신 메시지가 전송상에서 분실될 경우, 멤버들은 그룹 데이터의 복호화가 불가능해진다. 그러므로 신뢰성 있는 그룹키 갱신 메시지의 전달 뿐 아니라, 분실된 키의 복구는 매우 중요한 문제이다. 본 논문에서는 키 갱신 메시지의 분실이나 멤버의 로그인시에 발생할 수 있는 문제점을 분석하고 효율적으로 그룹키 및 보조키들을 복구하는 방법들을 제시한다. 또한 그룹 관리자가 메시지를 저장하지 않고 임의의 그룹키를 복구할 수 있도록 하였으며, 멤버의 중복된 키 복구 요청과 불필요한 보조키들의 전송을 줄였다.

1. 서 론

그룹 통신에서 가장 중요시되고 있는 보안 문제는 기밀성으로서, 허용된 사용자만이 그룹 데이터에 접근할 수 있도록 공유한 그룹키를 이용하여 암호화 통신을 한다[1]. 합법적 멤버만이 안전하게 그룹키를 공유하도록 하려면, 멤버가 가입하거나 탈퇴할 경우에 중앙의 제어자(GC: Group Controller)가 키를 갱신하고 이를 멤버들에게 전송한다. 만약 멤버가 키 갱신 메시지를 분실하면, 갱신된 그룹키를 얻을 수 없기 때문에 그룹 데이터를 복호화할 수 없을 뿐 아니라, 이후에 전송되는 키 갱신 메시지도 복호화가 불가능할 수도 있다. 이러한 키 갱신의 신뢰성에 관한 문제는 해결해야 할 문제로 논의되고 있으나[2][3], 이에 대한 연구는 미비하다. 현재는 키 갱신 메시지의 수신율을 높이기 위한 몇몇 방안들이 제시되었으나, 분실되었을 때 복구하는 방법에 대해서는 제시된 바가 없다. 본 논문에서는 통신의 지연이나 멤버의 로그 아웃으로 인하여, 분실된 메시지의 복구 이전에 키 갱신이 발생한 경우에도 효율적으로 키를 복구할 수 있는 방안을 제시한다.

2. 관련 연구

2.1 키-트리

규모가 큰 그룹에서 GC가 모든 멤버들에게 새로운 키를 1:1로 전송하는 것보다 효율적으로 그룹키를 갱신하기 위한 여러 가지 방법들이 연구되었다. 그 중에서 현재 가장 널리 사용되는 방법은 키-트리로서, 1998년 [1]에서 처음 소개되었으며 [4]에서는 이진트리를 사용하였다. 트리의 각 노드에는 유일한 키가 할당되어 있고, 단말 노드는 멤버에게 1:1로 대응된다. 모든 멤버는 자신의 단말 노드로부터 루트에 이르는 경로상의 키들을 소유하며, 루트키는 그룹키이다(그림 1 참조). 새로 가입한 멤버가 소유하게 되는 키는 모두 갱신되어 기존의 키로 암호화되어 전송되고, 탈퇴한 멤버가 소유했던 키들도 모두 갱신되어 각각 자식 노드키들로 암호화되어 전송된다.

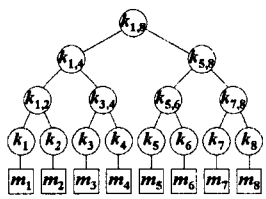

그림 1 이진 키-트리

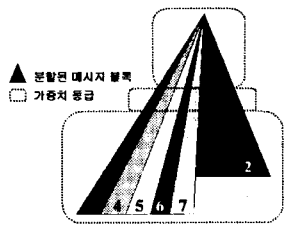

그림 2 메시지 구성 방법

2.2 신뢰성 제공을 위한 그룹키 관리

신뢰성이 처음 고려된 Keystone에서는 FEC (Forward Error Correction) 코드를 이용한 UDP 전송을 제안하였다[5]. [6]에서는 Keystone 방식이 연집 에러(burst error)에 대처하지 못함을 지적하면서, 키 갱신 메시지를 여러 개의 블록으로 분할하여 전송하는 방법을 제시하였다. 각 멤버에게 필요한 모든 키들은 1개의 블록에 포함되도록 구성하고, 각 블록마다 RSE (Reed Solomon Error) 코드를 계산하고 블록들을 중복 전송한다. [7]에서는 [6] 방식에서 키-트리의 상위 레벨에 있는 키일수록 많은 멤버들에게 공유된다는 사실을 감안하여, 레벨에 따라 몇 등급으로 나누어 가중치를 부여한다. 그리고 가중치에 따라 중복 전송하는 횟수를 늘리는 방식이다. (그림 2)에서 7개의 삼각형은 분할 메시지 블록을 나타내고, 점선으로 표시된 3개의 사각형은 서로 다른 가중치를 가지는 블록을 의미한다.

기술한 세 가지 방법들은 멤버들의 키 갱신 메시지 수신율을 높이는 것이 목표이다. 즉, 키 분실의 가능성을 완전히 배제할 수는 없으며 분실되었을 경우에는 재전송

† 본 논문은 한국전자통신연구원 네트워크보안연구부 위탁연구과제에 의한 것임.

을 반복한다. 그러나 그러한 재전송으로 지연이 발생한 상태에서 또 다른 키 갱신 이벤트가 발생할 경우에 대해서는 언급하지 않고 있다. 이러한 상황은 멤버가 로그아웃한 상태에서 버퍼링 되어 있어야 할 키 갱신 메시지가 분실되어 있을 때와 유사한 문제이다.

ELK에서[8] 멤버들은 분실된 키의 일부를 기존의 키로부터 직접 계산하고 나머지는 전수 조사하여 찾는다. 계산된 키의 검증값이 데이터 패킷과 함께 전달된 *hints*라는 값과 일치하면 올바른 킷값이다. 이것은 통신량을 절약하기 위해 고안된 방법으로서, 키를 갱신하기 위하여 전수 조사하는 멤버의 계산량이 많다.

3. 키 복구 메커니즘 제안

가장 기본적인 방법으로서, GC가 최근 w(복구 윈도우 크기)개의 최근 키갱신 메시지를 버퍼링하고 멤버의 복구 요청시 이를 재전송하는 것이다. 만약 버퍼에 존재하지 않으면, 복구 불가로 응답한다. 이 방법은 매우 간단하고 GC의 부가 처리 시간이 거의 불필요 하지만, w 이전의 키는 복구가 불가능하다. 또한 분실된 메시지 이후의 메시지들을 순차적으로 복호화하여야 하므로, 분실 이후 갱신되어 더 이상 사용되지 않는 보조키(KEK: Key Encryption Key)들도 복호화 하여야 한다.

3.1 용어 정의

- $event_i$: i번째 키 갱신을 유발시킨 멤버의 동작. *join* 혹은 *leave*.
- T_i: $event_i$로 인하여 i번째 갱신이 일어난 후의 키-트리.
- $ReKey_i$: T_{i-1}에서 T_i로 변경된 키를 멀티캐스트하기 위한 키 갱신 메시지.
- GK_i: $ReKey_i$에 의해 갱신된 그룹키로서, T_i의 루트키. i는 그룹키 버전.
- $level(node)$: 키-트리의 노드 $node$의 레벨. 루트의 레벨은 1이고, 각 노드의 레벨은 부모 노드의 레벨+1.
- $path_q^r$: T_r에서 m_q의 단말 노드로부터 루트까지의 경로.
- l_q^r : T_r에서 멤버 m_q와 $event_r$을 유발시킨 멤버 m_x와의 NCA(Nearest Common Ancestor) 키. 즉, $path_q^r$과 $path_x^r$가 만나는 첫 노드.
- $PRF_{key}(s)$: key를 키로 하여 s를 난수로 매핑하는 의사 난수 생성 함수[9].

3.2 분석 사항

개선된 키 복구 방법을 설계하기 위하여, 키-트리의 특성과 앞에서 기술했던 기본적인 방법을 사용할 경우의 문제점을 분석한 결과 다음과 같은 사항이 관찰되었다.

(1) 키-트리의 어떤 노드키가 $event_i$로 인하여 갱신되었다면, 이 노드의 선조 노드키들도 함께 갱신되었다.

(2) $ReKey_r$을 분실한 멤버 m_q는 다음과 같은 경우에 $GK_r,...,GK_{s-1}$ $(r<s)$를 복구하지 못한다:

그림 3은 $path_q^k$에서 멤버 탈퇴로 인한 GK_k 갱신 메시지인 $ReKey_k$에 포함된 키들을 표시한 것이다. 멤버의 탈퇴로 인한 키 갱신 메시지는 경로 상에서 변경되지 않은 최상위(레벨값이 가장 작은) 키에서 시작하여, 차상위 키를 암호화하는 체인을 형성한다. 멤버의 가입으로 인한 키 갱신 메시지는 체인을 형성하지 않고, 각 새로운 키들은 동일한 노드의 이전 킷값으로 암호화된다. 그림 3에서 보는 바와 같이 $GK_{r+1},GK_{r+2},GK_{r+3}$ 을 위한 키 체인은 분실된 $ReKey_r$에서 갱신된 키부터 시작하기 때문에 복호화가 불가능하다. 한편, $ReKey_{r+4}$의 키 체인은 $ReKey_r$의 분실과 상관없이 복호화할 수 있다. $l_q^s=l_q^r$ 일 경우에는 $event_s$가 *join*인지 *leave*인지에 따라 $ReKey_s$의 복호화 가능 여부가 달라진다. $event_s$가 *leave*였다면 $ReKey_s$의 키 체인은 l_q^r의 자식 노드키로 시작되므로 복호화가 가능하지만, $event_s$가 *join*이었다면 $ReKey_s$의 키들은 $ReKey_r$에서 갱신된 키들로 암호화되었기 때문에 복호화가 불가능하다.

(3) $ReKey_s$가 전송되었다면, $ReKey_r,...ReKey_{s-1}$ 에 포함된 KEK 들은 더 이상 키 갱신 메시지에 사용되지 않는다(그림 3 참조).

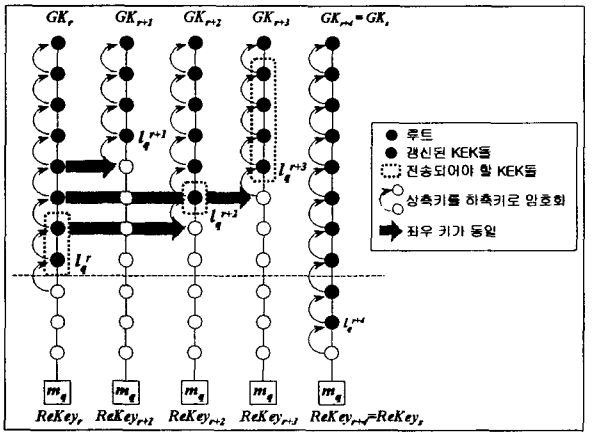

그림 3 멤버의 탈퇴로 인한 키 갱신 체인

3.3 제안 메커니즘

GC가 임의의 시간에 임의의 버전의 그룹키를 생성할 수 있도록 $GK_i=PRF_{Mkey}(i)$와 같이 생성한다. $Mkey$는 GC만 알고 있는 비밀키값이고, 그룹키는 GK_1부터 시작하여 갱신될 때마다 버전이 1씩 증가한다. 적절한 w를

설정하고 키-트리의 각 노드는 $last$와 크기가 w인 배열 $flag$를 가진다. $last$는 해당 키의 최근 변경을 발생시킨 그룹키 버전이고, $flag[i\%w]$는 이 키의 갱신을 유발한 $event_{i\%w}$를 저장한다. 멤버가 $ReKey_r$의 복구를 요청하면, GC는 다음 프로시저를 통하여 멤버가 복호화할 수 없는 최소의 그룹키들과 필요한 KEK들만을 전송한다.

단계	세부 단계	
윈도우 이전의 메시지일 경우	1.1	$GK_r, GK_{r+1}, ..., GK_c$ (c는 현재 그룹키 버전)를 메시지에 추가한다.
	1.2	$path_q^c$의 모든 키를 메시지에 추가한다.
윈도우 내의 메시지일 경우	2.1	루트로 올라가면서 l_q^r을 찾는다. 이 과정에서 단말 노드부터 l_q^r 사이에 $level(l_q^r) \le level(l_q^s)$인 최소 s를 찾는다. $level(l_q^r) = level(l_q^s)$ 일 경우에는 $leave$로 인한 키 갱신이어야 한다. 또한 이 때 만난 노드들의 $flag[(last+1)\%w], ..., flag[c\%w]$ 값을 지운다.
	2.2	만약 l_q^r가 루트이면 GK_c만 메시지에 추가한다.
	2.3	만약 l_q^r가 루트가 아니면 $GK_r, GK_{r+1}, ..., GK_{s-1}$ 를 메시지에 추가한다. 만약 조건을 만족하는 s가 존재하지 않으면, $ReKey_r$에 포함되지 않은 최상위 노드키부터 루트의 자식 노드키까지 메시지에 추가한다. 또한 이 노드들의 $flag[(last+1)\%w], ..., flag[c\%w]$ 값을 지운다.
메시지 전송	3.1	생성한 메시지를 m_q의 비밀키로 암호화하여 유니캐스트한다.

4. 안전성 및 효율성

그룹키는 GC만이 알고 있는 $Mkey$를 키로 하여 PRF를 이용하여 계산된다. 계산된 값은 하위 노드키로 암호화하여 전송되므로, 본 스킴의 안전성은 PRF와 암호화 알고리즘의 안전성에 의존한다.

재전송 방법은 저장한 메시지를 추가의 처리과정 없이 송신하므로 GC의 추가 수행시간 측면에서 제안한 방법보다 효율적이나 w 이전의 키는 복구할 수 없다. 제안한 방법에서 GC의 추가 저장 공간은 약 $2wK \cdot \log n$에서 $2w \cdot \log n$ 비트로 감소한다. 재전송 방법에서 메시지 크기는 멤버 가입과 탈퇴일 경우 각각 약 $K \cdot \log n$와 $2K \cdot \log n$이고, 메시지 분실 이후 발생한 키갱신의 수를 p라 할 때 제안한 방법의 메시지는 약 $K(p+\log n)$ 비트이므로, 평균적으로 $p < \log n$일 때 효율성이 증가한다. 또한 멤버의 계산량은 약 $p \cdot \log n$에서 $p+\log n$ 으로 현격히 감소한다.

5. 결론 및 향후 연구 과제

그룹 통신의 보안을 위해 사용되는 그룹키의 변경 메시지를 분실할 경우, 그룹 데이터의 복호화가 불가능하다. 그러므로, 신뢰성 있는 그룹키 갱신 메시지의 전달 뿐 아니라 분실된 키의 복구는 매우 중요한 문제이다. 현재 신뢰성 제공 방안에 대한 연구는 많지 않을 뿐더러, 그러한 연구들은 최근 키 갱신 메시지 수신의 실시간성을 주기 위한 방법에 대한 연구로서 메시지의 분실을 완전히 방지하지는 못한다. GC가 전송된 메시지를 저장하고 이를 재전송한다면, GC의 많은 저장 공간을 요구할 뿐 아니라 메시지를 분실한 멤버에게는 더 이상 유용하지 않은 키들을 전송하기 때문에 멤버의 계산량을 증가시키며, 저장되지 않은 메시지는 복구가 불가능하다. 본 논문에서는 이러한 문제점을 분석하고 효율적으로 그룹키 및 보조키들을 복구하는 방법을 제시하였다. 제안한 방법에서는 복호화가 불가능한 최소한의 그룹키 및 보조키들을 계산하고 전송한다. 또한 메시지를 저장하지 않으면서 임의의 그룹키를 복구할 수 있도록 하였다.

제안한 방법은 [5], [6]나 [7] 등의 방법을 먼저 적용하여 절절한 수신율을 유지하는 것이 GC로의 복구 요청 폭주를 방지하는 방법일 것이다. 향후 효율성 극대화를 위한 적절한 수신율과 $w(\ge 0)$ 설정에 대한 실험과 분석이 필요하다.

참고문헌

[1] C. K. Wong, M. Gouda, S. S. Lam, "Secure Group Communications using Key Graphs," Proc. of ACM SIGCOMM, Vol.28, Issue 4, Oct. 1988.

[2] http://www.ietf.org/html.charters/msec-charter.html

[3] http://www.securemulticast.org/msec-index.htm

[4] D. Wallner, E. Harder R. Agee, "Key Management for Multicast: Issues and Architectures," RFC 2627, June 1999.

[5] C. K. Wong & S. Lam, Keystone: A Group Key Management Service, Proc. of International Conference on Telecommunications, May 2000.

[6] X. Brian Zhang, S. S. Lam, D. Lee and Y. R. Yang, "Protocol Design for Scalable and Reliable Group Rekeying," Proc. of SPIE Conference on Scalability and Traffic Control in IP Networks, Aug. 2001.

[7] S. Setia, S. Zhu & S. Jajodia, A Scalable & Reliable Key Distribution Protocol for Group Rekeying, Technical report, George Mason Univ, Jan. 2002.

[8] A. Perrig, D. Song and J. D. Tygar, "ELK, a New Protocol for Efficient Large-Group Key Distribution," 2001 IEEE Symposium on Security and Privacy, May 2001.

[9] O. Goldreich, S. Goldwasser, and S. Micali, "How to Construct Random Functions," Journal of the ACM, Vol.83, Issue 4, Oct. 1986.