브로드캐스팅 통신 환경 하에서의 비상태 수신자를 위한 대역폭 효율성을 고려한 탈퇴 기법

(A Bandwidth-Efficient Revocation Scheme for Stateless Receivers in Broadcasting Communication Environment)

김 평 † 허 준 범 † 윤 현 수 ††
(Pyung Kim) (Junbeom Hur) (Hyunsoo Yoon)

요 약 Complete Subtree(CS)[1] 기법은 브로드캐스트암호화기법 중 비상태성을 만족시키는 부분집합-커버(Subset-Cover) 접근방법으로 그룹의 키의 배포 및 업데이트를 수행하는 방법이다. 브로드캐스트가 서비스 제공자로부터 단방향으로 이루어지는 비상태성 수신자를 가지는 환경에서는 키 관련 데이터 전송을 위해 저장 오버헤드와 통신 오버헤드가 발생한다. 본 논문에서는 효율적인 통신 오버헤드를 위해 새로운 탈퇴 기법인 Merged Complete Subtree(MCS) 기법을 제안한다. MCS기법에서는 CS기법에서 키 암호화 키(Key Encryption Key)를 분배하기 위해 사용하는 부분집합의 구성을 단순 트리구조가 아닌 합집합 기반의 복합 구조를 취한다. 이 수정 사항은 늘어난 전체 부분집합과 그에 해당되는 키 암호화 키로 인해 보다 많은 저장 오버헤드를 필요로 하지만 CS기법에서 헤더(header)를 구성하기 위한 두 부분집합을 하나의 합집합 형태로 표현 가능하게 함으로써 기존의 CS기법의 통신 오버헤드를 절반으로 감소시킨다. 제안된 기법은 전체 잠재적 사용자에 대한 탈퇴한 사용자의 비율이 커졌을 때 큰 이득을 가지며 공모를 통한 공격에 대하여 완벽한 저항성을 가진다.

키워드 : 브로드캐스트 암호화, 탈퇴 기법, 그룹 커뮤니케이션, 비상태성 사용자

Abstract Complete Subtree scheme(CS) is a well known broadcast encryption scheme to perform group rekeying in a stateless manner. However, statelessness comes at a cost in terms of storage and message overhead in transmitting key material. We propose a Merged Complete Subtree scheme (MCS) to reduce the communication overhead. It is more practical to make broadcast encryption schemes in network environments with limited bandwidth resources. We define all possible subset unions for ever two subsets of CS as new subsets having own key. The modification causes more storage overhead. Nevertheless, it is possible to make the size of a header, including key materials, half using subset unions of MCS, because the size of a header depends on the number of used subsets. Our evaluation therefore shows that the proposed scheme significantly improves the communication overhead of CS, reducing by half the rekey communication cost. The proposed scheme has the advantage of rekey communication cost when the number of revoked users is significant percentage of the number of potential users. The proposed scheme is fully collusion resistant.

Key words : broadcast encryption, revocation scheme, group communication, stateless receiver

· This research is supported by the Ubiquitous Computing and Network (UCN) Project, Knowledge and Economy Frontier R&D Program of the Ministry of Knowledge Economy(MKE) in Korea as a result of UCN's subproject 10C2-T1-20S.

† 학생회원 : KAIST 전산학과
pkim@nslab.kaist.ac.kr
jbhur@nslab.kaist.ac.kr
†† 종신회원 : KAIST 전산학과 교수
hyoon@nslab.kaist.ac.kr

논문접수 : 2009년 8월 6일
심사완료 2010년 5월 18일

정보과학회논문지: 정보통신 제37권 제5호(2010.10)

1. 서 론

최근 인터넷과 정보통신기술의 발달로 인해 상업적 목표를 가진 디지털 컨텐츠의 배포가 증가하고 있다. 이러한 예로 유료TV 및 각종 스트리밍 서비스 등을 들 수 있으며 회원제(membership)를 기반으로 한 서비스를 제공하기 위해서는 허가된 사용자들에게만 주어진 컨텐츠(contents)에 접근할 권한을 주는 기술이 필요하다. 이 목적에 부합되는 기술이 브로드캐스트암호화기법(Broadcast Encryption)[2]이며 각 사용자의 현재 세션에서의 회원 가입여부를 관리하기 위하여 암호작성법을 이용한 기술을 사용한다.

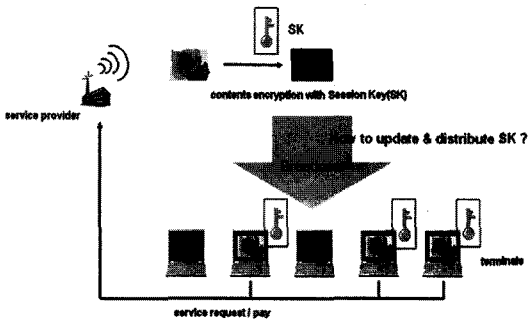

그림 1 브로드캐스트암호화기법

그림 1은 브로드캐스트암호화기법의 도식도이다. 그림과 같이 서비스 제공자는 세션 키(Session Key, SK)를 사용하여 컨텐츠를 암호화하고 합법적인 절차(서비스 가입/대금 지불)를 통하지 않은 사용자의 접근을 차단하며 인가 받은 특정 사용자에게만 세션 키를 공유하는 방식으로 서비스에 대한 접근을 허용한다. 이를 위해 세션 키는 현재 세션에 대해서 사용자의 서비스 가입여부에 의해 주기적인 업데이트가 이루어져야 한다. 즉, 브로드캐스트암호화기법은 세션 키를 관리하고 업데이트(update)하며 철회(revocation), 배포(distribution)하는 알고리즘이며 서비스 제공자와 사용자가 세션 키에 대한 정보를 주고 받기 위해 사용자와 서비스 제공자가 공유하고 있는 키 암호화 키(Key Encryption Key, KEK)가 있어 이를 사용하여 세션 키에 대한 정보를 암호화하여 교환한다.

IETF MSEC framework은 특정 그룹 커뮤니케이션(group communication)에 필요한 키를 관리하기 위한 절차(group key management protocols)를 정의하며 주기적인 업데이트, 철회, 배포를 위하여 GCKS(Group Controller and Key Server)를 정의한다. 일반적으로 키 암호화 키를 이용하여 세션 키를 암호화한 이후에 GCKS에서 브로드캐스트를 통해 그룹의 전체 사용자에게 세션 키를 전달하는 방식을 취하며 다양한 형태의 그룹 커뮤니케이션에서의 키 관리를 다루는 연구가 존재한다[1,3-11].

그룹 커뮤니케이션의 형태 중에서 사용자의 참여 및 탈퇴가 동적(dynamic)으로 매우 신속히 이루어짐으로 인해 서비스 제공자로부터 이루어지는 데이터 전송이 단방향통신의 특성을 띄는 상황을 가정해 볼 수 있다. 이러한 상황에서는 서비스 제공자와 사용자의 쌍방향의 교신이 이루어지지 않으므로 서비스 제공자는 사용자의 현 상태(state, 오프라인 off-line / 온라인 on-line)를 알 수 없고 과거에 전송된 키 관련 자료의 저장여부를 판별할 수 없다. 이때 서비스 제공자 측에서 사용자의 상태를 정의할 수 없는 비상태성(stateless)을 가진다고 하며 과거의 데이터 전송에 대하여 보관 여부를 판별할 수 없기 때문에 현재의 세션 키에 대한 정보는 데이터 전송이 이루어질 때마다 권한을 가진 사용자들만이 접근할 수 있는 형태로 함께 전송하여야 한다. 이를 위하여 서비스 제공자의 GCKS에서는 각각의 권한을 가진 사용자들과 공유하는 키 암호화 키를 사용하여 세션 키를 암호화한 후 취합하여 데이터의 헤더(header)로 구성한다. 이 데이터 헤더는 전송이 이루어 질 때 지속적으로 포함되어야 하므로 네트워크 대역폭 차원에서의 통신 오버헤드(communication overhead)가 발생하고 사용자와 서비스 제공자의 GCKS가 어떻게 헤더의 구성에 필요한 키 암호화 키를 공유하는 가에 있어서 저장 오버헤드(storage overhead)가 발생한다. 이러한 오버헤드를 최소화하는 것이 비상태성 사용자를 위한 그룹 커뮤니케이션을 수행하는 기법들의 목표라고 할 수 있으며 이는 주로 키 암호화 키를 어떤 형태로 구성하는가에 의해 큰 영향을 받는다(전체 '키 암호화 키'의 양이 증가할수록 사용자와 서비스 제공자가 공유해야 하는 키의 양이 증가하므로 저장 오버헤드가 증가하지만 이것은 사용자 사이에서 키 암호화 키의 공유를 증가시킴으로 인해 세션 키의 암호화 횟수를 감소시키므로 통신 오버헤드의 감소로 이어진다).

처음으로 사용자의 비상태성을 정의하고 이러한 상황에서 적용 브로드캐스트암호화기법을 다룬 연구[1]에서는 트리구조(tree)를 이용하여 키 암호화 키를 구성하는 CS기법(Complete Subtree scheme)과 SD기법(Subset Difference scheme)의 기본적인 두 가지 방법을 제안하고 있으며 이러한 키 암호화 키를 잠재적인 사용자를 포함한 전체 사용자의 부분집합에 할당시키는 방식의 접근법을 취하는데 이러한 방식을 부분집합-커버(Subset-Cover) 접근법이라고 부른다. 이러한 부분집합-커버 접근법과 CS/SD기법에 대해서는 2장에서 보다 자세히 설명한다. 이외에 부분집합-커버 접근법은 [1]에

서 제시한 CS/SD기법을 기반으로 저장 오버헤드와 통신 오버헤드를 개선시키는 형태로 관련 연구가 이루어졌다. D. Halevy, A. Shamir는 LSD기법을 제안하여 SD기법에서 사용되는 전체 부분집합 사이의 계층 구조를 이용하여 같은 사용자에 대한 적용범위를 가지는 중복적인 부분집합을 제거하는 방식으로 저장 오버헤드에 대한 개선이 이루어졌다[4]. T. Asano와 R. Nojima, Y. Kaji의 경우 키를 생성하는 방식(Key Generation)의 변화를 통하여 사용자와 GCKS 사이에서 보다 효율적으로 저장소를 사용할 수 있는 방법을 제안하고 있다[3,5]. 이외에 통신 오버헤드를 줄이기 위한 방식으로 동적 구조(dynamic structure)를 이용하거나[8] 이중적인 트리 구조를 사용하는 방법[9], 사용자의 비상태성을 인정하지 않는 기법의 방법론을 SD기법에 도입하는 방법[7] 등이 제안되었지만 이러한 통신 오버헤드를 줄이는 방법들은 엄밀하게 사용자의 비상태성을 완벽하게 만족시키지는 못하였다.

본 논문에서는 통신 오버헤드를 줄이는 것에 중점적인 목표를 두며 사용자의 완벽한 비상태성을 충족시키는 브로드캐스트암호화기법을 제안하고자 한다. 이를 위해 CS기법과 SD기법의 차이점을 분석하고 새롭게 키 암호화 키를 구성하는 구조체를 정의할 것이다. 본 논문은 다음과 같이 구성된다. 2장에서는 기존 CS기법과 SD기법에 대해서 살펴본다. 3장에서는 2장에서의 분석을 바탕으로 새로운 기법을 제안한다. 그리고 제안된 기법에서의 키 생성방식에 대한 설명을 다루고 이를 보완한다. 4장에서는 제안된 기법의 안전성을 살펴보고 이를 분석한다. 5장에서는 성능에 대하여 기존의 기법과 비교분석한다. 마지막으로 6장에서는 논문을 요약, 결론 맺는다.

2. 관련 연구

브로드캐스트암호화기법 중에서 사용자의 비상태성에 대한 조건을 만족시키기 위한 일반적인 형태가 언급된 부분집합-커버 접근법을 사용한 것이며 [1]에서 소개되고 있다. 이를 위해서는 잠재적인 사용자를 모두 포함하는 집합 N이 있다고 할 때 $|N| = N$이고, 미리 정의된 부분집합의 컬렉션(collection) S가 존재하여 $S=\{S_1, S_2, \cdots\}$, $S_i \subseteq N$을 만족한다. 각각의 부분집합 $S_i \subseteq N$에 대하여 K_i를 할당하고 S_i 포함되는 사용자가 아니라고 하면 K_i를 알 수 없도록 한다. 이때 K_i는 GCKS와 부분집합에 포함된 사용자간에 사전 공유되어야 하는 키 암호화 키가 된다. 새로운 세션 키(SK)를 배포할 때는 현재 허가 받지 않은 사용자(revoked user)의 집합을 $R(|R| = r)$이라 했을 때 GCKS에서는 다음을 만족하는 부분집합-커버 S'를 구한다.

$$S' = \{S_{i_1}, S_{i_2}, \cdots\} \subseteq S,\ \cup S' = S_{i_1} \cup S_{i_2} \cup \cdots = N \setminus R$$

이후 GCKS는 다음과 같이 세션 키에 관련된 키 메시지 헤더를 구성하여 배포한다.

$$S', E_{K_{i_1}}(SK), E_{K_{i_2}}(SK), \cdots$$

여기서 S'는 어떤 부분집합에 해당되는 K_i를 사용했는가를 나타내어 주기 위하여 사용자에게 색인(index) 역할을 하는 용이한 표현법이 되고 $E_K(\ )$의 경우 K를 키로 사용하는 AES와 같은 대칭키(symmetry key)방식의 암호화 알고리즘이다. 전체 부분집합의 컬렉션 S는 고정되어 있고 변화하지 않으므로 최초의 공유 이후 사용자에게 별도의 키 암호화 키인 S_i에 해당하는 K_i의 업데이트를 요구하지 않는다. 따라서 이런 형태의 브로드캐스트암호화기법은 사용자의 비상태성을 만족시킨다. 이러한 부분집합-커버 접근법은 어떠한 방식으로 부분집합을 구성하고 각각에 키를 할당하였는지에 따라 부분집합-커버를 구하기 위한 방법, 사용되는 컬렉션에 포함되는 부분집합의 숫자, 공유되어야 하는 부분집합의 키(: 키 암호화 키)의 숫자가 달라진다. 다음은 그 중 기본적이고 대표적인 CS/SD기법의 부분집합 컬렉션의 구성방법에 대한 설명과 이를 개선시킨 방식의 설명이다.

2.1 Complete Subtree 기법

CS기법은 부분집합-커버 접근법의 한 가지 기법으로 모든 사용자를 말단(leaf)에 할당하는 이진트리(binary tree)를 구성하고, 이진트리의 노드(node) v를 루트(root)로 하는 서브트리(subtree)의 모든 말단을 S_v로 표시 한다. 그럼 이때 모든 S_v에 대하여 $S_v \in S_{CS}$가 성립하는 최소한의 S_{CS}가 존재하고 이를 CS의 부분집합의 컬렉션으로 둔다. 그림 2는 S_2과 S_6에 대한 부분집합과 키 메세지 헤더의 예제이다.

$$Msg: \{S_2, S_6\}, E_{K_{S_2}}(SK), E_{K_{S_6}}(SK)$$

CS기법의 경우 노드 8에 할당되는 사용자 8이 저장해야 키는 사용자8이 포함되는 부분집합 S_8, S_4, S_2, S_1에 관련한 키가 되므로 O(log N)이 되고, 키 메시지 헤더의 크기는 O(r · log N/r) 이 된다[1].

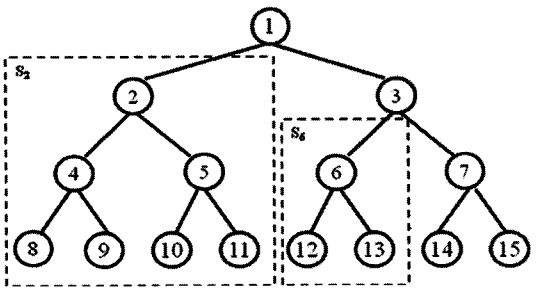

그림 2 Complete Subtree

2.2 Subset Difference 기법

SD는 CS방식의 부분집합 컬렉션이 주어진다고 했을 때, 임의의 두 부분집합의 차(difference)를 부분집합으로 표시하는 방식으로 부분집합의 컬렉션을 구성한다. (모든 사용자가 말단으로 할당 되는 이진트리에서), 노드 v를 루트로 하는 서브트리의 모든 말단 중에서 w를 루트로 하는 서브트리의 말단을 제거한 부분을 $S_{v,w}$로 표시하고 모든 $S_{v,w}$에 대하여 $S_{v,w} \in S_{SD}$가 성립하는 최소한의 S_{SD}를 SD의 부분집합 컬렉션으로 둔다. 그림 3는 $S_{2,10}$과 $S_{6,12}$에 대한 부분집합과 키 메세지 헤더의 예제이다.

$$Msg : \{S_{2,10}, S_{6,12}\}, E_{K_{S_{2,10}}}(SK), E_{K_{S_{6,12}}}(SK)$$

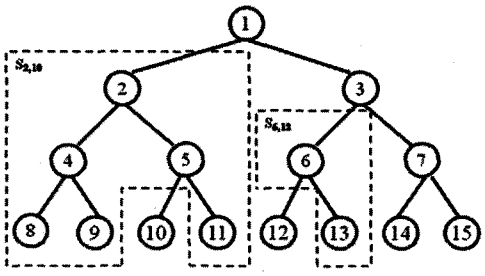

그림 3 Subset Difference

SD기법의 경우 노드 8에 할당되는 사용자8이 저장해야 키는 사용자8이 포함되는 부분집합 $S_{4,9}$, $S_{2,5}$, $S_{2,9}$, $S_{2,10}$, $S_{2,11}$, $S_{1,3}$, $S_{1,5}$, $S_{1,6}$, $S_{1,7}$, $S_{1,9}$, $S_{1,10}$, $S_{1,11}$, $S_{1,12}$, $S_{1,13}$, $S_{1,14}$, $S_{1,15}$에 관련한 키가 되므로 $0(\log^2 N)$ (: pseudo-random generator 사용시 $0(\log N)$)이 되고, 키 메시지 헤더의 크기는 $0(\min(2r-1, N-r))$ (평균적으로 1.38r)이 된다[1].

2.3 CS기법의 개선

2.2에서 언급된 CS기법을 개량한 기법 중 대표적인 것은 [3]에서 제안하고 있는 단방향 트랩도어 퍼뮤테이션 함수(one-way trapdoor permutation function)를 이용하여 키를 생성시키는 방법이다.

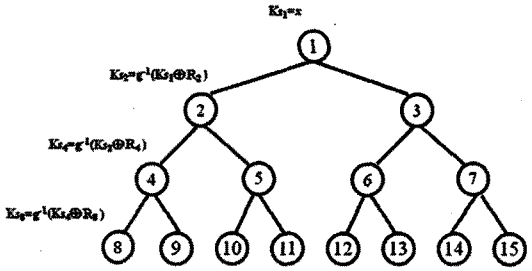

그림 4 단방향 트랩도어 퍼뮤테이션 – g 를 이용한 키 생성 과정

그림 4는 사용자8이 가져야 하는 키의 발생 방법이다. 임의의 노드 i가 있을 때, R_i는 각 노드에 할당된 트랩도어 정보(trapdoor information)이고 이 값은 특정 해쉬 함수(hash function)를 이용하여 생성할 수 있는 공개된 값이다. 그리고 g^{-1}은 단방향 트랩도어 퍼뮤테이션 함수 – g에 대하여 $x = g^{-1}(g(x))$을 만족시키는 역함수(invert function)이며 g는 공개되어 있지만 g^{-1}은 구하기 힘들다고 가정한다. 그리고 g^{-1}은 GCKS만이 알고 이를 사용해서 키를 생성한다. 위와 같은 방식으로 키를 생성할 때, 사용자8은 K_{S_8}만을 보관하면 g와 각 노드의 트랩도어 정보를 이용해서 자신이 포함된 모든 부분집합에 대한 키를 유도하는 것이 가능하다. 따라서 사용자가 가져야 하는 정보의 저장 오버헤드는 O(1)으로 수렴하게 된다. 이 방식은 CS기법의 저장 오버헤드를 줄이지만 실제 사용되고 있는 부분집합과 키 암호화 키의 구조는 변하지 않았으므로 CS기법의 통신 오버헤드는 변함이 없다[3].

2.4 SD기법의 개선

2.3에서 언급된 기법 이외에 SD기법을 개량한 방법 중 대표적인 것으로 [4]에서 제안한 LSD 기법을 들 수 있다. LSD기법은 SD기법에서 사용한 부분집합의 다음과 같은 특성을 이용한다.

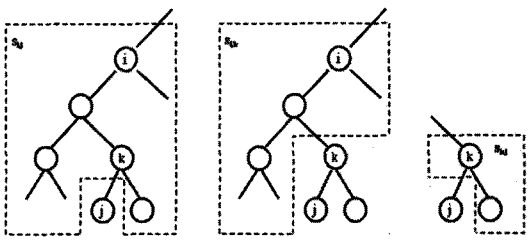

그림 5 SD기법에서 사용하는 부분집합의 특성

그림 5에서 루트에서부터 말단까지 이어지는 경로에 i, k, j가 순서대로 있다고 한다. 그때 $S_{i,j} = S_{i,k} \cup S_{k,j}$이 성립한다는 것을 확인할 수 있다. 이러한 부분집합 사이의 특성을 이용하여 LSD기법에서는 기존의 SD기법의 부분집합 $S_{i,j}$에서 i와 j를 취할 수 있는 레벨을 전체 레벨이 아니라 특정 레벨(special level)을 포함하는 형태로 제한하고 이외의 부분집합은 합집합 형태로 재구성해서 사용하도록 제한을 둔다. 따라서 사용자가 저장해야 하는 키가 줄어들기 때문에 저장 오버헤드가 줄어든다. 그러나 그림 5에서의 예시와 같이 동일한 사용자들에게 세션 키를 전송하기 위해 $S_{i,j}$의 키 암호화 키를 사용할 경우 하나의 키를 사용한 암호화만이 필요한 반면 $S_{i,k}$와 $S_{k,j}$가 사용된 경우 2개의 키 암호화 키를 사용한 2번의 암호화 과정이 필요하고 이것은 통신 오

버헤드의 증가를 의미한다. 일반적으로 SD기법과 비교해 LSD기법은 저장 오버헤드는 O(log(N)·log log(N))으로 줄어드는 반면 통신 오버헤드는 O(r·log log(N))으로 증가한다[4].

2.5 CS기법과 SD기법의 비교

2.2와 2.3에서 CS기법과 SD기법을 살펴보았고 이후에 이를 개선한 두 가지 기법 또한 살펴보았다. 그러나 기존의 CS기법과 SD기법과 비교해 볼 때 각각의 기법들은 주로 저장 오버헤드의 개선에 초점이 맞추어져 있으며 LSD기법의 경우는 오히려 통신 오버헤드는 trade-off로 증가했음을 확인할 수 있다. 이들 기법에 대하여 통신 오버헤드에 초점을 두고 비교해 볼 때 몇 가지 공통점과 차이점을 발견 할 수 있다. 첫 번째로 통신 오버헤드와 저장 오버헤드의 관계이다. 2.1에서 언급하였듯이 부분집합-커버 접근법에서는 현재 상태에서 권한을 가진 사용자만을 포함할 수 있는 최소한의 부분집합-커버 - S'를 구해야 하며 이때 구해진 S'에 대해서 |S'|(: 포함하고 있는 부분집합의 수)에 의해 키 메시지 헤더의 길이 즉, 통신 오버헤드가 결정된다. S'에 포함되는 부분집합이 적을수록 각각의 부분집합의 키 암호화 키를 사용한 세션 키의 암호화 작업이 감소하고 결과적으로 키 메시지 헤더의 길이가 줄어든다. 적은 수의 부분집합으로 현재 권한을 가진 사용자를 효율적으로 포함하는 S'를 구한다는 것은 GCKS가 가지고 있는 부분집합의 다양성이 요구됨을 의미한다. 그리고 이러한 부분집합의 S의 다양성은 사용자와 GCKS가 공유하고 있는 부분집합 $S_i \subseteq S$와 해당 K_i의 숫자가 증가한다는 것을 의미하므로 저장 오버헤드와 통신 오버헤드는 근본적인 trade-off 관계에 있음을 설명할 수 있다.

두 번째 고려할 상황은 CS기법과 SD기법에서 구해지는 부분집합-커버 S'의 |S'|는 1/2 N을 초과할 수 없다는 것이다. CS의 경우 1/2 N명의 사용자가 서비스를 제공받을 경우 구해지는 |S'|는 아래의 그림 6과 같은 상황에서 1/2 N이 되며, 이와 같은 상황에서 사용자 증감에 상관없이 1/2 N 보다 증가할 수 없다.

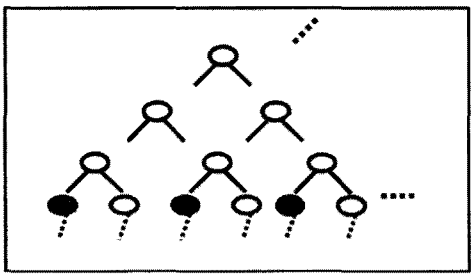

그림 6 트리구조기반 부분집합-커버 알고리즘의 비효율적인 상황

SD기법의 경우 탈퇴한 사용자의 수 r에 대하여 부분집합-커버의 가능한 최대 크기가 2r-1로 구해지는데 이 값이 1/2 N을 초과할 경우에는 부분집합-커버를 구하는 것이 아니라 현재 세션에 참여하고 있는 N - r < 1/2 N 만큼의 각각의 개인에게 주어진 개별적인(individual) 키를 이용해서 키 메세지 헤더를 작성할 수 있다. 그림 7은 서비스 탈퇴자의 증가에 따른 통신 오버헤드의 변화의 upper bound를 그래프로 나타낸 것이다.

이 그래프에 SD기법의 경우 1/2 N 보다 r이 적은 상황에서 대부분의 경우 SD기법이 CS기법에 비해 네트워크 대역폭의 사용 측면에서 유리하지만 탈퇴한 사용자의 수 r이 증가할수록 점차 통신 오버헤드가 CS기법을 초과하여 증가한다 것을 확인할 수 있다. 이 점에 착안하여 3장에서는 CS를 수정하여 r의 값이 N과 비교하여 상당한 비율을 차지하는 경우에 유리할 수 있는, 또는 GCKS가 실사용자에게 키를 배분한 시기가 오래지 않

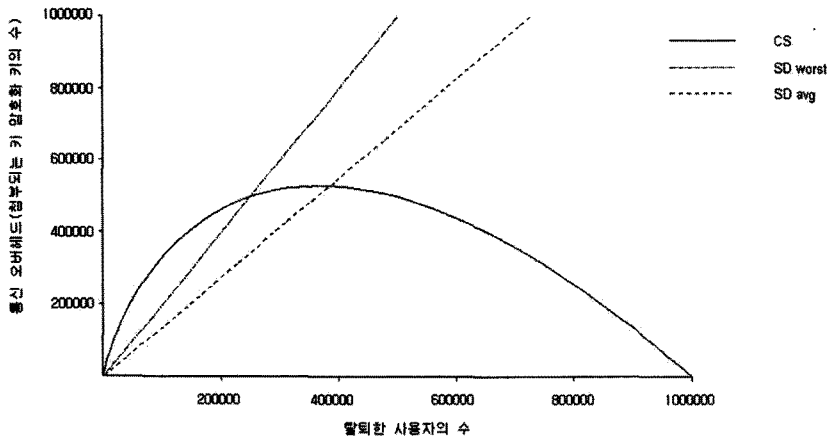

그림 7 CS 방식과 SD 방식의 네트워크 대역폭 성능 비교

은 기간 즉, 서비스제공자가 시스템을 시작한 초기 상황에 보다 효율적으로 작동할 수 있는 MCS(Merged Complete Subtree)기법을 제안한다.

3. MCS기법

2장에서 언급한 바와 같이 전체 부분집합 컬렉션 S의 크기와 통신 오버헤드는 서로 trade-off 관계에 있다. 제안하려고 하는 기법은 이러한 특성을 이용하여 부분집합의 컬렉션을 구성할 때 CS기법과 비교하여 제곱에 해당하는 다양성을 갖도록 설계하여 네트워크 대역폭의 효율성을 높이도록 설계한다.

3.1 The Framework

제안하고자 하는 MCS기법의 부분집합 컬렉션 M의 구성은 CS의 부분집합 컬렉션인 S_{CS}의 개념을 차용하여 정의하도록 한다.

정의 1.

$$S_{CS} = \{S_1, S_2, ..., S_{2N-1}\},$$
$$\exists i,j,\ M_{ij} = S_i \cup S_j,\ S_i \not\subset S_j,\ S_i \not\supset S_j \text{ 일 때,}$$
$$\forall M_{ij} \in M \text{을 만족하는 최소한의 } M\text{을 MCS의}$$
부분집합의 컬렉션으로 둔다.

즉 MCS에서 사용하는 부분집합의 컬렉션 M은 CS기법의 부분집합 컬렉션 S_{CS} 에서 임의의 포함관계가 없는 두 S_i, S_j의 합집합들의 모임이라고 말할 수 있다. 각각의 M_{ij}에 대응하는 키를 MK(i,j)라 하고 M_{ij}에 포함되는 모든 사용자 즉, S_i와 S_j에 포함되는 모든 사용자와 GCKS가 공유한다. 이때 각각의 S_i 와 S_j에 대해서는 키를 할당하지 않는다.

이러한 MCS를 통한 키 메세지 헤더의 작성과 해독은 CS기법과 흡사하게 이루어지지만 각각 CS기법에서 사용되는 부분집합을 해당되는 M_{ij}로 치환하는 과정이 들어간다. 브로트캐스트 과정 iii 과 해독 과정의 ii 과정

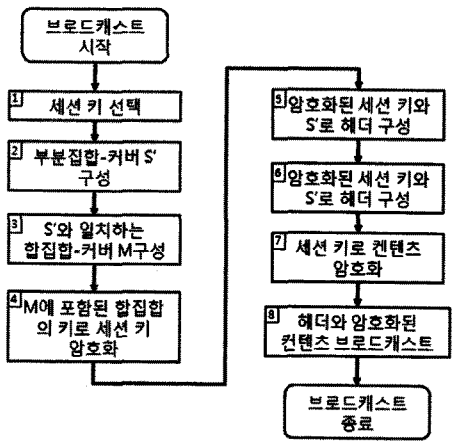

그림 8 브로드캐스트 과정

이 이를 포함하고 있다.

그림 8은 브로드캐스트 과정의 도식도 이다. 브로드캐스트 과정에서는 GCKS는 임의의 세션 키를 정하고 현재 허가된 사용자들을 커버할 수 있는 가장 최소한의 부분집합-커버 S'를 구한다. 이 과정에서 사용되는 각 부분집합은 CS기법에서 사용하는 것과 동일하다. 이후 해당 S'에서 사용된 부분집합을 정의 1의 합집합 형태의 부분집합으로 두 개씩 치환하고 이에 해당되는 합집합의 키를 키 암호화 키로 삼아 세션 키를 암호화 하고 암호된 세션 키와 세션 키를 사용하여 암호화시킨 컨텐츠와 함께 브로드캐스트한다. 아래의 i 과정이 도식도의 1)번에 해당하고 ii 과정이 도식도의 2)번, iii 과정이 3)4)번, iv 과정이 5)6)7)8)번 에 해당한다.

브로드캐스트 과정 (GCKS) :

i) 세션 키 SK를 선택한다.

ii) 주어진 N과 R에서 권한을 가진 사용자만을 포함하는 최소한의 부분집합-커버 $S' = \{S_{i_1}, S_{i_2}, ..., S_{i_{2m-1}}\}$ 또는 $S' = \{S_{i_1}, S_{i_2}, ..., S_{i_{2m}}\}$을 찾는다.

iii) S'의 각 부분집합을 연속하는 2개의 쌍으로 구분하여 각각의 부분집합에 해당되는 M의 부분집합의 키, $MK(i_1, i_2)$, $MK(i_3, i_4)$, ..., $MK(i_{2m-1}, t)$, 또는 $MK(i_{2m-1}, i_{2m})$을 구한다. (이때, t는 구해진 S'의 m이 홀수 일 때를 대비한 가상의 부분집합이다. GCKS는 가상의 사용자 1명을 상황에 맞게 온라인/오프라인 시킨다고 가정한다.)

iv) 에서 구한 키들을 이용하여 다음과 같은 키 메세지 헤더를 포함한 암호문(ciphertext)을 작성한다.

$$[S', E_{MK(i_1,i_2)}(SK), E_{MK(i_3,i_4)}(SK), ...,\ ]\ F_{SK}(P)$$

이때, 이 암호문을 제공받는 사용자 측에서는 어떠한 키 암호화 키와 세션 키를 사용하였는지 알 수 없으므로 다음과 같은 메시지를 전송 받는다고 볼 수 있다.

$$[S', E_{MK(i_1,i_2)}(SK), E_{MK(i_3,i_4)}(SK), ...,\ ]\ F_{SK}(P)$$
$$= [S', C_1, C_2, ...,\ ],\ P'$$

F는 세션 키를 이용해서 단시간 안에 효율적으로 컨텐츠 P를 암호화하는 알고리즘이다.(e.g. stream cipher algorithm)

그림 9는 해독 과정의 도식도이다. 해독 과정에서는 전해 받은 암호문은 아래의 i 과정의 그것과 같고, 여기서 사용된 부분집합-커버에 대한 정보를 S'로 부터 추출하여 자신이 관련된 부분집합을 구한다. 이후 어떤 부분집합과의 합집합으로 재구성 되었는지를 판별하여 세션 키를 해독하고, 해독한 세션 키를 이용해서 컨텐츠를 해독할 수 있다. i 과정이 도식도 1), ii 과정이 도식도 2)3)4), iii 과정이 도식도 5)에 해당한다.

해독 과정(사용자) :

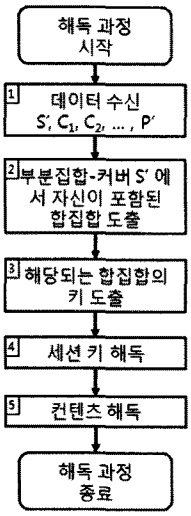

그림 9 해독 과정

i. S_{i_j} 에 포함되는 사용자는 다음과 같은 전체 데이터를 수신받고

$$[S', C_1, C_2, ...,], P'$$

S'에서 자신이 포함된 S_{i_j}를 확인한다.

ii. 1에서 i_j를 확인했을 경우 $C\left\lfloor \frac{j}{2} \right\rfloor$ 과 자신이 소유한 $MK\left(i_{\left\lfloor \frac{j}{2} \right\rfloor \cdot 2-1}, i_{\left\lfloor \frac{j}{2} \right\rfloor \cdot 2}\right)$을 이용해서 해독하여 세션 키 $SK = D_{MK\left(i_{\left\lfloor \frac{j}{2} \right\rfloor \cdot 2-1}, i_{\left\lfloor \frac{j}{2} \right\rfloor \cdot 2}\right)}\left(C\left\lfloor \frac{j}{2} \right\rfloor\right)$를 구한다. 이때, $D_K(\ )$는 $x=D_K(E_K(x))$를 만족하는 $E_K(\ )$의 역을 구하는(해독) 알고리즘이다.

iii. 세션 키 SK를 이용하여 컨텐츠 $P = F^{-1}_{SK}(P')$를 구한다.

F^{-1}는 $x = F^{-1}_{SK}(F_{SK}(x))$를 만족하는 F의 역을 구하는 알고리즘이다.

3.2 키 생성 절차

제안된 MCS기법에서는 CS기법에서의 임의의 두 부분집합의 합으로 부분집합을 구성하므로 GCKS에서 보관해야 하는 부분집합 컬렉션 M의 부분집합에 해당하는 키의 수는 $|M| = (2N-1)(2N-\log N)/2$가 되고 사용자가 보관해야 하는 키의 수는 $(\log N-1)(2N-\log N)$이 된다. 이러한 오버헤드는 CS기법의 서버 $|S_{CS}| = 2N-1$, 사용자 $\log N$와 비교해 보았을 때, 무척 크다고 할 수 있고 이를 줄이기 위해서, 단방향 퍼뮤테이션 함수(ony-way permutation function)[6]를 이용하여 키를 생성하고 차후 4장에서 이에 대한 안정성 여부를 살펴보도록 한다.

그림 10과 같이 말단을 각각의 사용자에게 할당시킨

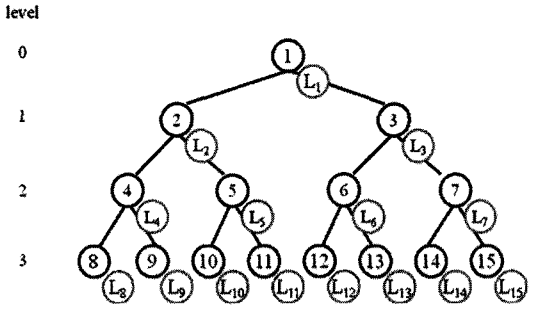

그림 10 트리 구조 기반의 난수 라벨

트리 구조체가 있다고 할 때 각각의 노드 i에 라벨(Label) $L_i \in \{0, 1\}^\lambda$ 라는 난수 값을 기준 시드(seed)와 키를 생성하기 위해 사용하도록 한다. λ는 보안 매개변수(security parameter)이다. 그리고 주어진 시드 값을 퍼뮤테이션하기 위한 함수를 단방향 퍼뮤테이션 함수 $f_L, f_M, f_R : \{0, 1\}^\lambda \rightarrow \{0, 1\}^\lambda$ 로 정의한다. L_i는 f_L, f_M, f_R을 이용해 퍼뮤테이션 된 값만이 키와 시드로 사용자들에게 공개되고 L_i에 대한 값은 공개되지 않는다.

M_{ij}는 $S_i \cup S_j$ 로 가정할 경우, i와 j가 같은 레벨일 경우일 때는 $MK(i,j)=f_M(L_j \oplus L_k)$을 키로 사용한다. 그리고 i, j가 다른 레벨일 경우, 일반성을 잃지 않고 i<j 라고 했을 때 노드 k는 i의 조상 중에서 j와 같은 레벨에 있는 노드로 둔다. 이때, M_{ij}의 키인 MK(i, j)는 L_j와 L_k를 이용하여 생성된 시드로 생성되게 된다. 이때 시드는 j<k일 경우 $f_R(f_L(L_j) \oplus L_k)$이 되고 k<j일 경우 $f_L(f_R(L_j) \oplus L_k)$인 형태를 취하게 된다. 이후 노드 k에서부터 노드 i에 이르는 경로에 일치하도록 오른쪽으로 진행할 경우 f_R, 왼쪽으로 진행할 경우 f_L을 이용해서 퍼뮤테이션을 진행해 나간다. 그리고 노드 i에 이르렀을 때, 값을 f_M을 이용해 다시 퍼뮤테이션 시킨 값이 MK(i,j)가 된다. 다음은 L_2와 L_3를 이용한 시드와 키의 유도 과정이다

예를 들어, 위의 그림 11에서 사용자 u_5가 가져야 하는 키 중에서 S_2와 관련된 키는 MK(2,3), MK(2,6), MK(2,12) 3개가 되고 이 값들은 S_2에 포함된 u_1, u_2, u_3, u_4는 모두 공유해야 하는데 시드값인 $f_R(f_L(L_2) \oplus L_3)$과 MK(2,3)만을 저장하는 것으로 유도 가능하게 된다. 더 나아가 $f_R(f_L(L_2) \oplus L_3)$과 MK(2,3) 값은 S_2와 관련된 S_3의 모든 부분집합의 합집합에 해당되는 키에 대해서도 유도가 가능하게 되므로 u_1, u_2, u_3, u_4 는 저장소의 사용을 줄일 수가 있다. 그러나 $f_R(f_L(L_2) \oplus L_3)$ 값은 S_3에 포함되는 사용자에게는 노출이 되어선 안되다. 만일 u_5가 이 값을 가질 경우 이 시드를 이용하여 MK(2,13)과 같이 자신이 포함되지 않는 부분집합에 대한 키를 유도해 낼 수 있기 때문에 S_3에 포함된 사용자들에게는 해당되는 키를 직접 저장해야 하는 필요가 있다. 그림

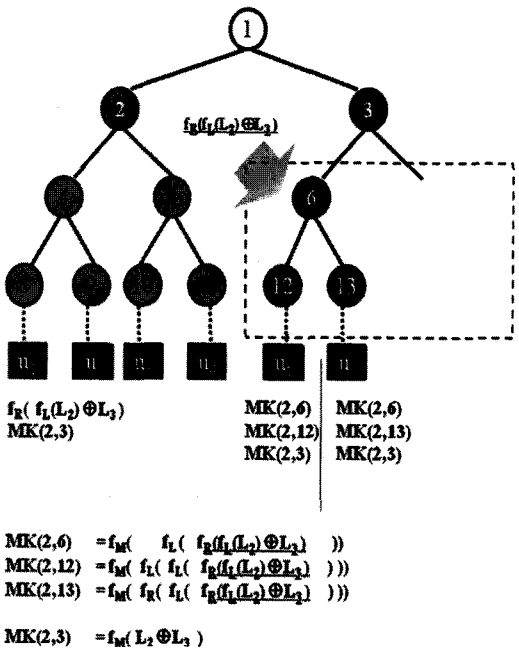

그림 11 S_2의 사용자와 S_3의 각 부분집합에 포함된 사용자에 대한 키 생성 과정

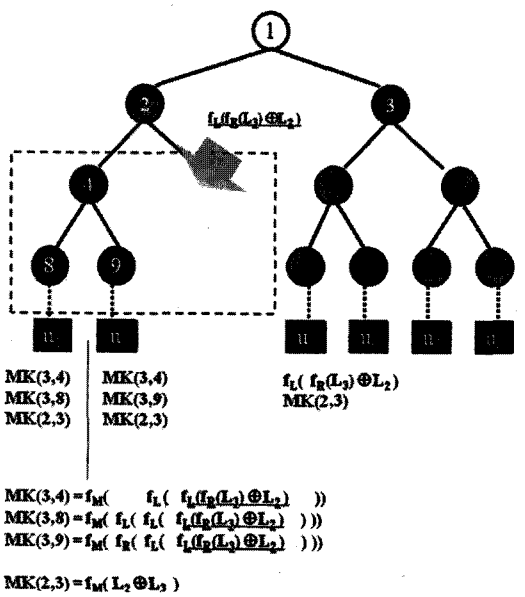

그림 12 S_3의 사용자와 S_2의 각 부분집합에 포함된 사용자에 대한 키 생성 과정

12는 $f_L(f_R(L_3)\oplus L_2)$을 이용하여 u_1, u_2가 가져야 하는 키를 생성하는 예시이다.

이와 같은 방식으로 키를 생성할 경우 GCKS가 저장해야 하는 값은 2N - 1로 줄어들고 이 값 또한 생성함수(generating function)를 이용할 경우 O(1)까지도 줄이는 것이 가능하다. 기존의 사용자가 저장해야 했던 (log N - 1)(2N - log N)의 키의 양도 다음과 같은 계산을 통해 O(N)으로 줄어든다는 것을 알 수 있다.

$$\sum_{i=0}^{\log N}\{(2^i-1-i)+2\cdot(2^i-1)\}-(N-1)$$
$$=5N-0.5\log^2 N-3.5\log N-5$$

위의 수식은 특정 사용자가 각 레벨에서 가져야 하는 키와 시드의 양을 계산한다. 2^i-1-i는 특정 사용자가 i레벨에서 자신이 포함되는 서브트리의 루트를 기준으로 한 부분집합과 다른 부분집합과의 합집합에 해당되는 키 중에서 키 형태로 저장해야 하는 양이 되고 이는 i 미만의 레벨의 노드 중 기준 노드의 조상을 제외한 것들의 개수와 동일하다. $2\cdot(2^i-1)$는 기준 노드와 같은 레벨의 노드에 해당되는 라벨들을 이용해서 생성해야 할 시드의 숫자이고 이는 현재 레벨의 노드 중에서 자신을 제외한 개수와 동일하다. 그리고 마지막 레벨의 경우에는 라벨 값을 이용해서 시드를 생성할 필요가 없기 때문에 N-1을 감산하여 준다.

4. 안전성 분석

브로드캐스트암호화기법에서 만족시켜야 하는 보안기준(security criteria)은 사후기밀성(forward confidentiality), 사전기밀성(backward confidentiality), 그리고 공모저항성(collusion resistance)이 있다. 사후기밀성과 사전기밀성은 세션의 회원관리와 연관된 보안기준이다. 회원을 탈퇴한 사용자가 이후 세션에서 제공되는 컨텐츠에 접근할 수 없도록 하는 것이 사후기밀성이며 현재 세션에서 인가된 사용자들이라고 하더라도 이전에 인가되지 않았던 세션의 컨텐츠에 접근할 수 없도록 하는 것이 사전기밀성이 된다. CS기법과 SD기법은 구조상으로 새로운 사용자의 가입을 허용하고 있지는 않기에 사전기밀성에 대한 언급을 할 수 없지만 사용자의 서비스 가입 이전 상태, 자신의 키 암호화 키를 할당 받지 않은 상태를 실제 구현상으로 오프라인으로 처리할 경우, 이전 세션 키의 암호화에 사용된 키 중에서 해당 사용자의 키 암호화 키는 배제 되므로 구현 상으로(practically) 사전기밀성을 보장한다고 말할 수 있다. 그리고 CS기법과 SD기법은 현재 세션에 참가하고 있는 사용자들에 대하여 정확한 부분집합-커버를 구하므로 탈퇴한 사용자에 대해서 사후기밀성을 보장한다. 마지막으로 공모저항성은 탈퇴한 사용자나 공격자들이 모의를 통해 현재 세션의 컨텐츠 혹은 세션 키에 대한 접근이 가능한가를 살피는 문제이다. CS기법과 SD기법의 경우 사용되는 모든 키가 랜덤수열과 구분할 수 없는 키-구분불가능성(key-indistinguishability)을 만족하므로 공모저항성을

완벽히 만족한다고 언급하고 있다[1].

제안된 MCS기법의 경우 CS기법과 동일한 사용자 포함영역을 갖는 부분집합-커버를 사용하여 세션 키를 암호화하기 때문에 사후기밀성과 사전기밀성에 대해 CS기법과 동일한 안전성을 갖는다고 볼 수 있다. 그러나 공모를 통한 공격에 대한 안전성 여부를 확인하기 위해서는 키를 생성하는 방식과 단방향 퍼뮤테이션 함수의 안전성 등 몇 가지 사항에 대해서 자세히 논의할 필요가 있다.

공모를 통한 공격에 대한 안전성을 확인하기 위하여 관련연구 [3]에서 제시한 또 다른 안전성의 기준인 키-난해성(key-intractability)을 차용하고자 한다. 키-구분불가능성은 보다 강한 안전성의 기준으로 임의의 키가 주어졌을 경우에 난수와 구분할 수 없는 경우이며 키-난해성은 공격자가 임의의 키에 대해서 다항시간(polynomial-time)에 이를 찾아낼 수 없는 경우에 대한 확률적인 모델이 된다.

정의 2. (단방향 퍼뮤테이션[6]) : 특정 인덱스(index)의 집합 $I \subseteq \{0, 1\}^*$ 에 대하여 모든 퍼뮤테이션 함수의 집합을 $F = \{ f_i : D_i \rightarrow D_i \mid i \in I \}$로 표기하기로 할 때 다음을 가정할 수 있다.

i. 효율적인 표본추출 알고리즘(sampling algorithm) S가 존재하여 특정 인덱스 i와 $x \in D_i$를 생성할 수 있다.

ii. f_i는 공개된 함수이므로 효율적으로 $y=f_i(x) \in D_i$를 계산할 수 있다.

iii. 확률적 알고리즘(probabilistic algorithm) A가 f_i를 역을 취하는데 있어 A가 얻을 수 있는 이득은 다음과 같다 :

$$Adv^1_{A,F}(\lambda) = \Pr\{x = x' \mid x,i \leftarrow S,\ y = f_i(x), x' \leftarrow A(i,y)\}$$

$t(\lambda)$ 시간 동안 $\varepsilon(\lambda)$보다 높은 확률로 A가 F의 함수를 역을 취할 수 있을 때 A는 $(t(\lambda), \varepsilon(\lambda))$-break F라고 표현하며, 이러한 A가 존재하지 않을 때, F는 $(t(\lambda), \varepsilon(\lambda))$-secure하다고 표현한다. 확률적 알고리즘 A가 가질 수 있는 이득 $Adv^1_{A,F}(\lambda)$이 λ에 대해 무시할만(negligible) 할 때 F의 퍼뮤테이션 함수 f_i는 역을 취하기 어렵다고 이야기 할 수 있다.

이러한 정의 2를 바탕으로 다음과 같은 보조정리 1을 이끌어 낼 수 있다.

보조정리 1.

$$Adv^1_{A,F}(\lambda) = \Pr\{x = x' \mid x,i \leftarrow S,\ y = f_i(x), x' \leftarrow A(i,y)\} < \varepsilon(\lambda)$$

이고 $f_{i_1}, f_{i_2}, \ldots, f_{i_m} \in F$가 있을 때, $f_{i_m}(f_{i_{m-1}}(\cdots(f_{i_1}(x))\cdots)) = y$를 역을 취하기 위한 확률적 알고리즘 A'가 가질 수 있는 이득 $Adv^2_{A,F}(\lambda)$ 또한 $Adv^2_{A,F}(\lambda) < \varepsilon(\lambda)$을 만족한다.

증명. 증명은 귀납적 방법에 의해 이루어진다. f_1, f_2에 대해 $z = f_1(y)$, $y = f_2(x)$이 성립한다면 공격자가 얻을 수 있는 이득은 먼저 $z = f_1(y)$에 대하여 성공적으로 y를 구한 경우와 그렇지 못한 경우를 생각해야 한다. 만일 y를 구해내지 못하였다면 $y = f_2(x)$에서 z를 구해낼 확률은 전체 도메인에서 임의의 한 값을 선택할 확률과 같다. 전체 계산은 다음과 같다.

$$\Pr\{x = x' \mid y = f_2(x),\ x' \leftarrow A(2,y) \cap z = f_1(y),\ y' \leftarrow A(1,z),\ y = y'\} + \Pr\{x = x' \mid y = f_2(x),\ x' \leftarrow A(2,y') \cap z = f_1(y),\ y' \leftarrow A(1,z),\ y \neq y'\} = \Pr\{x = x' \mid y = f_2(x),\ x' \leftarrow A(2,y) \times \Pr\{y = y' \mid z = f_1(y),\ y' \leftarrow A(1,z)\} +$$

$$\frac{1}{\| D_2 \|}(1-\varepsilon(\lambda)) < \varepsilon^2(\lambda) + \frac{1}{\| D_2 \|} < \varepsilon(\lambda)$$

따라서 $f_{i_m-1}(f_{i_{m-2}}(\cdots(f_{i_1}(x))\cdots)) = t$에서의 공격자가 얻을 수 있는 이득이 $\varepsilon(\lambda)$에 bound된다면 같은 방법으로 $f_{i_m}(f_{i_{m-1}}(\cdots(f_{i_1}(x))\cdots)) = s$에서 공격자가 얻을 수 있는 이득 또한 $\varepsilon(\lambda)$에 bound 된다는 것이 성립함을 확인할 수 있다.

제안된 MCS기법에서 특정 i, j에 대하여 지정했을 때, S_i와 S_j에 포함되지 않은 모든 사용자들이 가지는 키 관련 정보(유도된 키 또는 시드)를 $\overline{K}(i, j)$로 둔다. 그리고 MCS에 대한 공격자 C가 있어서 $\overline{K}(i, j)$의 정보를 이용해서 다항 시간에 MK(i, j)를 구해낼 확률을 $Adv^3_{C,F}(\lambda)$이라고 하자. 그리고, 보안 매개변수 λ에 대한 다항식(polynomial) ω가 존재하여 $N = \omega(\lambda)$이 성립한다고 두면, $Adv^3_{C,F}(\lambda)$가 어떠한 공격자와 ω에 대해서도 무시할 만하다면 제안된 MCS기법은 키-난해성을 가지고 있다고 할 수 있다.

MCS에서 트리 구조는 실제 저장되지는 않지만 가상적으로 CS에서 사용하는 트리 구조와 같이 말단에 각 사용자를 할당한 완벽한 이진트리구조(full binary tree)라 생각하기로 하고, 노드 i를 루트로 하는 서브트리의 말단으로 이루어지는 부분집합을 S_i, 노드 a가 노드 b의 조상일 경우 $a < b$로 표시하기로 한다.

다음은 공격자 C가 퍼뮤테이션 함수 f_R, f_L, f_M에 대한 공격을 통하여 MK(i, j)를 알아내는 공격 과정 분석을 통하여 제안된 MCS기법은 이 공모를 통한 공격에 대하여 안전하다는 것을 살피는 과정이다. (만일 C가 f_R, f_L, f_M에 대한 공격이 아니라 다른 방식으로 MK(i, j)를 알아내기 위해서는 퍼뮤테이션 결과와 난수를 구분해내는 키-구분불가능성 문제가 된다.)

정리 1. 퍼뮤테이션 함수 f_R, f_L, f_M가 단방향 퍼뮤테이션 함수일 때, MCS는 공모를 통한 공격에 대해 안전하다(fully-collusion resistant).

증명. MK(i, j)가 MCS기법의 부분집합 M_{ij}의 키라고 둔다.

MK(i, j) = MK(j, i)이므로 일반성을 잃지 않고 i > j라고 둘 수 있다.

i) $\lceil \log i \rceil = \lceil \log j \rceil$: 노드 i와 j의 레벨이 같을 경우

MK(i, j) = $f_M(L_i \oplus L_j)$

공격자가 $\overline{K}(i, j)$을 알고 있을 때, MK(i, j)을 구하기 위해서는 $L_i \oplus L_j$을 구해야만 가능하고 이 값은 다음과 같은 방법으로 구할 수 있다.

∃노드 t, $\lceil \log t \rceil = \lceil \log i \rceil = \lceil \log j \rceil$,

a. $\exists f_M(L_i \oplus L_t)$, $\exists f_M(L_j \oplus L_t) \in \overline{K}(i, j)$ 이거나

b. $\exists f_L(f_R(L_t) \oplus L_i)$, $\exists f_L(f_R(L_t) \oplus L_j) \in \overline{K}(i, j)$ 이거나

→ $j < i < t$ 일 때 만들어지는 시드

c. $\exists f_R(f_L(L_t) \oplus L_i)$, $\exists f_R(f_L(L_t) \oplus L_j) \in \overline{K}(i, j)$ 인 경우

→ $t < j < i$ 일 때 만들어지는 시드

$$L_i \oplus L_j = f_M^{-1}(f_M(L_i \oplus L_t)) \oplus f_M^{-1}(f_M(L_j \oplus L_t))$$
$$= f_L^{-1}(f_L(f_R(L_t) \oplus L_i)) \oplus f_L^{-1}(f_L(f_R(L_t) \oplus L_j))$$
$$= f_R^{-1}(f_R(f_L(L_t) \oplus L_i)) \oplus f_L^{-1}(f_R(f_L(L_t) \oplus L_j))$$

이외의 경우에는 위의 과정을 유도하기 위한 중간 과정이거나 모든 Label L이 공개된 값이 아니므로 구할 수 없다. 따라서 가능한 t의 경우를 계산해보면

a. i, j가 말단에 위치할 경우 가장 많이 생길 수 있다. $\omega(\lambda) - 2$

b. i, j가 말단의 parent일 경우 가장 많이 생길 수 있다. $\omega(\lambda)/2 - 2$

c. i, j가 말단의 parent일 경우 가장 많이 생길 수 있다. $\omega(\lambda)/2 - 2$

특정 i, j에 대해서 $\overline{K}(i, j)$를 이용하여 가능한 계산 방법은 최대 $2\omega(\lambda) - 6$ 가지 방법이 존재할 수 있고 각각은 2회씩 $f_i \in F$를 역을 취하는 문제이다. 따라서 이 경우의 C가 얻을 수 있는 이득은 $Adv^3_{C,F}(\lambda) < (4\omega(\lambda) - 24) \cdot Adv^1_{A,F}(\lambda) < \varepsilon(\lambda) \cdot (4\omega(\lambda) - 24)$가 된다.

(이 값이 $Adv^3_{C,F}(\lambda)$의 tight upper bound는 아니다.)

ii) $\lceil \log i \rceil \neq \lceil \log j \rceil$: 노드 i와 j의 레벨이 다를 경우

∃k, $k < i$, $\lceil \log k \rceil = \lceil \log j \rceil$: k는 i의 조상 중에 j와 레벨이 같은 노드이다.

a. ∃p, $p \neq k$, $k < p < i$, $p \neq i$인 노드 p에 대해서, MK(j, p)를 이용할 경우.

이러한 MK(j, p)를 유도하는 시드는 $f_R(f_L(L_j) \oplus L_k)$, 또는 $f_L(f_R(L_j) \oplus L_k)$이 되는데 이 값은 S_j에 속하는 사용자에게만 알려져 있고 $\overline{K}(i, j)$에는 포함되지 않는다. 각 MK(j, p)에서 위의 시드 또는 시드로부터 유도된 중간 퍼뮤테이션 결과값을 구하여 MK(j, i)를 유도할 수 있는데 이때, MK(j, p)에서 시드를 구하는 문제는 보조정리 1에 해당하는 $f_i \in F$의 합성함수를 역을 취하는 문제이다. 따라서 위에 해당하는 p의 수는 다음과 같이 계산할 수 있다.

$$\left(2^{\log(\omega(\lambda)) - \lceil \log k \rceil + 1} - 1\right) - \left(2^{\log(\omega(\lambda)) - \lceil \log i \rceil + 1} - 1\right) - 1$$
$$= 2 \cdot \omega(\lambda)\left(\frac{1}{2^{\lceil \log k \rceil}} - \frac{1}{2^{\lceil \log i \rceil}}\right) - 1$$

b. MK(i, j)를 유도하는 시드로부터 직접 구할 경우.

$f_R(f_L(L_j) \oplus L_k)$ 또는 $f_L(f_R(L_j) \oplus L_k)$을 구해야 하는데 각각 $f_L(L_j) \oplus L_k$, $f_R(L_j) \oplus L_k$는 $\overline{K}(i, j)$로 부터 다음과 같이 구할 수 있다.

∃노드 t, $\lceil \log t \rceil = \lceil \log k \rceil = \lceil \log j \rceil$,

$\exists f_R(f_L(L_j) \oplus L_t)$, $\exists f_M(L_k \oplus L_t) \in \overline{K}(i, j)$ → $j < t$일 때 만들어지는 시드

$$f_L(L_j) \oplus L_k = f_R^{-1}(f_R(f_L(L_j) \oplus L_t)) \oplus f_M^{-1}(f_M(L_k \oplus L_t))$$

$\exists f_L(f_R(L_j) \oplus L_t)$, $\exists f_M(L_k \oplus L_t) \in \overline{K}(i, j)$ → $t < j$일 때 만들어지는 시드

$$f_R(L_j) \oplus L_k = f_L^{-1}(f_L(f_R(L_j) \oplus L_t)) \oplus f_M^{-1}(f_M(L_k \oplus L_t))$$

가능한 t의 경우를 생각해보면 j, k, t가 말단의 parent일 때 이다. 이때 $j < t$ 와 $t < j$ 의 조건은 서로 대칭 이므로 i, j를 정할 때 두 경우의 합이 최대 $\omega(\lambda) - 2$가 되는 것을 알 수 있다. 그리고 각각 2회씩 $f_i \in F$를 역을 취하는 문제이다.

a, b에 의해서 ii)인 경우에 대한 공격자의 이득은

$$Adv^3_{C,F}(\lambda) = \left(2 \cdot \omega(\lambda)\left(\frac{1}{2^{\lceil \log k \rceil}} - \frac{1}{2^{\lceil \log i \rceil}}\right) - 1\right) \cdot Adv^2_{A,F}(\lambda)$$
$$+ 2 \cdot (\omega(\lambda) - 2) \cdot Adv^1_{A,F}(\lambda)$$
$$< \left(2 \cdot \omega(\lambda)\left(\frac{1}{2^{\lceil \log k \rceil}} - \frac{1}{2^{\lceil \log i \rceil}} + 1\right)\right) \text{ 이 된다.}$$

i), ii)에 대하여 (not tight) upper bound를 구하면 $Adv^3_{C,F}(\lambda) < 4 \cdot \omega(\lambda) \cdot \varepsilon(\lambda)$라고 말할 수 있고 이에 의해 $\omega(\lambda)$는 보안 매개변수에 의한 다항식이므로 주어진 f_R, f_L, f_M을 역을 취할 수 있는 확률과 이의 합성함수에 대한 역을 취할 수 있는 확률이 무시할 만 하다면 $Adv^3_{C,F}(\lambda)$ 역시 무시할 만 하다고 말할 수 있다.

5. 성능 분석

브로드캐스트암호화기법에서 성능 비교의 기준은 저장 오버헤드와 통신 오버헤드를 얼마나 효율적으로 줄일 수 있도록 설계하였는가 이다. 2.3에서 언급한 바와 같이 제안된 MCS기법은 근본적인 두 기준의 trade-off 관계를 인식하고 저장 오버헤드의 최소한의 증가를 통하여 네트워크 대역폭 사용의 효율성을 증대할 목적으로 설계 되었다.

제안된 MCS기법은 기존의 CS기법과 SD기법과 비교해 볼 때 저장 오버헤드가 증가하였다[1]. 표 1은 big-O 표현 방식으로 한 점근적 순서(asymptotic order)의 비교이다.

표 1 CS기법, SD기법, MCS기법의 성능 비교

		CS	SD	MCS
저장 오버헤드	GCKS	O(N)	O(N)	O(N)
	사용자	O(log N)	$O(\log^2 N)$	O(N)
통신 오버헤드		$r \log \frac{N}{r}$	$2r - 1$, (avg 1.38r)	$\frac{1}{2} r \log \frac{N}{r}$

3장에서 제시된 브로드캐스트 과정과 해독 과정에서와 같이 기존의 CS에서 키 메세지 헤더를 구성할 때 사용되는 부분집합-커버의 부분집합을 각각 2개씩 합쳐 M에 포함된 부분집합으로 부분집합-커버를 구성하므로 세션 키를 암호화 하는데 필요한 부분집합의 키가 CS의 1/2이 되고 키 메시지 헤더의 길이에서 가장 많은 비율을 차지하는 부분이 암호화된 세션 키의 양이기 때문에 통신 오버헤드 역시 CS의 1/2에 가깝게 줄어든다.

전체 잠재적 사용자의 수 N = 10^6 라고 하였을 때, 탈퇴한 사용자의 수 r의 증가에 따른 그래프는 다음과 같다.

부분집합-커버 방식을 이용하는 브로드캐스트암호화 기법은 일반적으로 r이 증가할수록 허가 받은 사용자를 커버하기 위하여 사용되어야 하는 부분집합의 수가 증가하고 각각의 부분집합에 해당되는 키로 세션 키를 암호화하여 헤더로 구성하므로 통신 오버헤드가 증가하게 된다. (r이 1/2N을 초과할 경우 각각의 개인 키로 세션 키를 보내는 것이 더 효율적이므로 고려하지 않는다.) 이때 r이 적은 경우에 대해서 SD기법이 가장 효율적인 통신 오버헤드를 갖게 되지만 r이 증가할수록 CS기법과 MCS기법이 보다 효율적인 통신 오버헤드를 가진다. MCS기법의 경우 항상 CS기법의 절반에 해당되는 통신 오버헤드를 가지므로 MCS기법이 이런 경우 가장 효율적임을 알 수 있고, 평균적으로 SD기법에서 r이 N의 약 15%이상 인 경우, SD기법이 최악의 성능을 보이는 경우에는 r이 N의 약 6% 이상을 차지하는 경우 MCS의 통신 오버헤드가 SD기법 보다 효율적이다.

6. 결 론

사용자의 비상태성을 만족시키기 위해 부분집합-커버 접근법을 사용하는 브로드캐스트암호화기법은 세션 키에 대한 정보를 헤더로 구성하여 컨텐츠를 전송할 때마다 같이 전송한다. 이때 세션 키를 암호화 하기 위해 서비스제공자와 사용자들이 사전에 공유해야 하는 키 암호화 키로 인한 저장 오버헤드가 발생하고 컨텐츠를 제공할 때 마다 세션 키에 대한 헤더가 추가로 전송이 이루어지므로 통신 오버헤드가 발생한다. 본 논문은 기존의 대표적인 부분집합-커버 접근법을 이용하는 CS기법의 수정을 통해 사후기밀성(forward confidentiality)을 만족시키고, 공모를 통한 공격에서 안전하며, CS기법의 1/2만큼의 통신 오버헤드를 요구하는 MCS기법을 제안하였다. 평균적으로 탈퇴자의 수(r)가 전체 잠재적 사용자의 수(N)의 6%이상을 차지할 경우 CS기법뿐만 아니라 SD기법에 대해서도 보다 효율적인 성능을 보인다. 제안된 MCS기법은 추가적인 저장 오버헤드가 요구되지만 이에 대한 제한이 없는 상황에서 탈퇴한 사용자의 수가 전체 잠재적 사용자의 수와 비교해 큰 비율을 차지하는 상황의 시스템을 구현하는데 있어 가장 적합하다.

참 고 문 헌

[1] D. Naor, M. Noar, J. Lotspiech, "Revocation and tracing schemes for stateless receivers," CRYPTO 2001, LNCS, vol.2139, pp.41-62, 2001.

[2] A. Fiat, M. Naor, "Broadcast encryption," *Advances in Cryptology - CRYPTO'93, LNCS,* vol.773, pp.480-491, 1994.

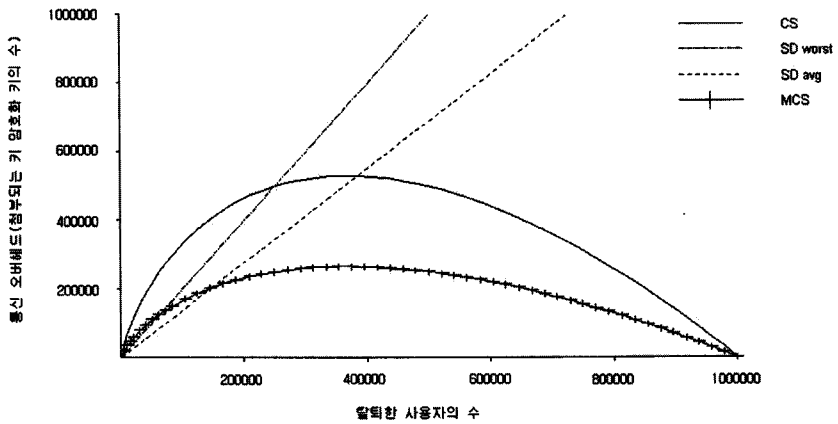

그림 13 MCS기법, CS기법, SD기법의 네트워크 대역폭 성능 비교

[3] R. Nojima, Y. Kaji, "Secure, Efficient and Practical Key Management scheme in the Complete-Subtree method," *IEICE TRANS. FUNDAMENTALS*, vol.E88-A, no.1, 2005.
[4] D. Halevy, A. Shamir, "The LSD broadcast encryption scheme," *CRYPTO'02, LNCS*, vol.2442, pp.47-60, 2002.
[5] T. Asano, "A revocation scheme with minimal storage at receivers," *ASIACRYPT'02, LNCS* vol.2501, pp.433-450, 2002.
[6] O. Goldreich, S. Goldwasser, S. Micali, "How to Construct Random Functions," *Journal of the ACM*, vol.33, no.4, pp.792-807, Oct 1986.
[7] M. Johansson, G. Kreitz, F. Lindholm, "Stateful Subset-Cover," *ACNS 2006, LNCS*, vol.3989, pp. 178-193, 2006.
[8] W. Chen, Z. Ge, C. Zhang, J. Kurose, D. Towsley, "On Dynamic Subset Difference Revocation scheme," *NETWORKING 2004, LNCS* vol.3042, pp.743-758, 2004.
[9] M. J. Mihaljevic, "Broadcast Encryption Schemes Based on the Sectioned Key Tree," *ICICS 2003, LNCS*, vol.2836, pp.158-169, 2003.
[10] C. K. Wong, M. G. Gouda, S. S. Lam, "Secure Group Communications Using Key Graphs," *ACM SIGCOMM*, pp.68-79, 1998. (in Canada)
[11] D.A. McGrew, A.T. Sherman, "Key Establishment in Large Dynamic Groups Using One-Way Function Trees," *Technical Report* No. 0755, TIS Labs at Network Associates, Inc., Glenwood, MD (1998).

김 평
2007년 한국과학기술원 전산학과 학사
2009년 한국과학기술원 전산학과 석사
2009년~현재 한국과학기술원 전산학과 박사과정, 관심분야는 네트워크 보안, 정보보안, 암호학

허 준 범
정보과학회논문지 : 정보통신
제 37 권 제 1 호 참조

윤 현 수
정보과학회논문지 : 정보통신
제 37 권 제 1 호 참조