제 18 회 한국정보처리학회 추계학술발표대회 논문집 제 9 권 제 2 호 (2002.11)

안전한 링크-상태 라우팅을 위한 확장된 보안 메커니즘의 제안

유병익*, 박창섭*, 최석용*
*단국대학교 전자계산학과
e-mail : youbi@cs.dankook.ac.kr

Extending the Security Mechanism for Secure Link-State Routing

Byung-Ick You*, Chang-Seop Park*, Seok-Yong Choi*
*Dept. of Computer Science, Dankook University

요 약

안전한 라우팅 프로토콜의 계산적 비용을 줄이기 위한 많은 제안들이 있었다. 본 논문에서는 이미 제안되었던 안전한 링크-상태 라우팅 기법을 소개하고, 안전한 링크-상태 라우팅 프로토콜에서 LSU(Link State Update) 프로세싱과 관련하여 기존의 문제점을 개선, 확장한 효율적인 보안 메커니즘에 대하여 기술한다.

1. 서론

인터넷에서 각각의 패킷은 발신지에서 목적지까지 경유되는 각각의 네트워크를 빠르게 통과해야 하기 때문에 네트워크 상에서 교환되는 라우팅 메시지(routing messages)는 인터넷의 가장 중요한 구성요소 중 하나이다. 현재 인터넷에서 가장 많이 사용되는 방법은 최단 경로를 따라 패킷이 전달되도록 설계된 것인데, 실제로 링크-상태 라우팅 프로토콜(link state routing protocol)로는 OSPF(Open Shortest Path First)[1] 그리고 거리-벡터 라우팅 프로토콜(distance-vector routing protocol)로는 RIP(Routing Information Protocol) [7] 등이 사용된다.

거리-벡터 라우팅 프로토콜을 수행하는 각각의 라우터는 최선의 경로를 찾기 위한 계산을 수행하고, 또한 계산과정이 진행되는 동안 인접 라우터와 라우팅 정보를 교환하여 라우팅 테이블을 유지한다. 반면에, 링크-상태 라우팅 프로토콜은 모든 라우터가 모든 목적지에 이르는 최단 경로를 계산하기 위해 전체 네트워크 형상(topology)을 유지하고, 각각의 라우터는 LSU(Link State Update)를 주기적으로 flooding 해준다.

이러한 기존의 라우팅 프로토콜들은 라우팅 정보를 보호하기 위한 보안 메커니즘을 기반으로 하고 있지 않기 때문에, 올바르게 프로토콜을 따르지 않는 라우터에 의해 전체 네트워크의 성능을 손상시킴으로써 악의적인 라우터 공격이 가해질 수 있다. 라우터에 대한 공격이 악의적이든 아니든 여러 유형의 공격으로부터 안전한 라우팅 프로토콜이 존재해야 하며 다음의 몇 가지 성질을 만족해야 한다. 첫째, 프로토콜은 올바르게 실행되어야 하고, 프로토콜이 손상되는 시점을 탐지할 수 있어야 한다. 둘째, 올바르게 작동하지 않는 라우터에 의해 발생하는 피해를 억제해야 한다. 셋째, 수신 메시지는 식별 가능한 호스트 또는 라우터로 부터 보내졌다는 것에 대한 확인을 할 수 있어야 한다. 넷째, 수신 메시지의 내용이 송신 메시지와 동일하다는 것을 확인 할 수 있어야 한다. 다섯째, 프로토콜을 수행하기 위해 주고받는 모든 메시지는 최신의 메시지(fresh message) 임을 확인하여 그로 인해 발생할 수 있는 재생 공격(replay attack)으로 부터 보호 될 수 있어야 한다.

지금까지 라우팅 보안(routing security)을 위해서 많은 방법들이 제안되었는데, R. Perlman [3] 은 공개키 기술을 기반으로 하여 hop-by-hop 단위로 서명을 생성

하고, 확인하는 방법을 제안하였고, K.Zhang [5] 은 일회용 서명(one-time signature)을 사용하는 방법을 제안했다. 그러나 공개키 기술을 기반으로 하는 서명의 생성과 확인은 라우팅 알고리즘에서 단순한 테이블 참조나 계산에 사용하기에는 많은 비용이 들고, 효율성이 떨어진다. 따라서, 라우팅 비용을 줄이기 위해 해쉬함수를 이용한 방법들이 R. Hauser et.al [2], S. Cheung [6], T. Goodrich [4] 등에 의해 제안되었다. 이 중에서 R. Hauser et.al [2] 의 방식은 링크-상태 값이 두 개(UP, DOWN)일 경우에는 효율적이지만, 링크-상태의 값이 늘어나면 라우팅 비용이 많이 들게 되는데, 본 논문에서는 [2]의 방식을 개선하여 안전한 링크-상태 라우팅을 위해 라우팅 비용을 최소화 시키는 방법에 관해 기술한다.

2. 기존방식

R. Hauser et.al [2]의 제안에서 라우터의 링크-상태가 안정적인 경우에 사용되는 SLS(Stable Link State) 기법과 링크-상태가 자주 변경되는 경우에 효과적인 FLS(Fluctuating Link State) 기법이 제안되었다.

이를 위해, RSA, DSS, ElGamal 등의 공개키 기반의 디지털 서명과 MD5, SHA 등의 일방향 해쉬함수를 기반으로 한 다음과 같은 해쉬체인(hash chain)을 사용하였다. 즉, r 을 임의의 난수(random number), H 를 일방향 해쉬함수라고 할 때, 다음과 같은 해쉬체인이 구성되어 질 수 있다.

$$H^1(r), \ldots, H^i(r), \ldots, H^t(r)$$

여기에서, t 는 해쉬체인의 길이, $H^i(r) = H(H^{i-1}(r))$, $1 \le i \le t$, 그리고 $H^0(r) = r$ 로 정의된다.

[2]에서는 LSU 의 유형을 두 가지로 분류하는데, 라우터의 수정, 삭제시 또는 해쉬체인이 모두 사용되었을 때 생성하여 분배되는 LSU 를 Anchor LSU (ALSU) 라 하고, 주기적으로 라우팅 정보를 교환할 때 분배되는 LSU 를 Chained LSU(CLSU)라 정의한다.

본 논문에서는 FLS 를 개선하고, 확장하여 동적 라우팅 환경에 적합한 방법을 제안할 것이므로 다음 절에서는 FLS 에 관해 기술한다.

2.1 Fluctuating Link State (FLS)

FLS 에서 라우터는 각각의 링크 L_j ($1 \le j \le l$) 마다 길이가 t 인 두 개의 해쉬체인을 생성하는데, 이 두 개의 해쉬체인은 링크-상태 값인 UP 과 DOWN 의 단지 2 가지 링크-상태에 대해서 각기 다른 해쉬함수를 사용하여 생성한다. 즉, 각각의 라우터는 해쉬 테이블 형태 (t행/l열)의 해쉬체인을 생성함에 있어, 임의의 난수 r_j 를 초기값으로 하여 모든 UP 체인은 해쉬함수 H 로 생성하고, 모든 DOWN 체인은 해쉬함수 G 로 생성한다.

링크 L_j 에서 생성되는 UP 상태와 DOWN 상태에 관한 해쉬체인은 다음과 같고, 이 해쉬체인을 사용하여 ALSU 와 CLSU 의 생성과 처리가 행해지게 된다.

UP 상태 : $H^1(r_j), H^2(r_j), \ldots, H^t(r_j)$
DOWN 상태 : $G^1(r_j), G^2(r_j), \ldots, G^t(r_j)$

[ALSU 의 생성 및 처리]

모든 초기값 r_j ($1<j<l$)는 공통된 하나의 r 로부터 생성될 수 있지만 유일한 값이어야 하고, ALSU 는 해쉬테이블의 계산으로 구성될 수 있으며, 현재 시간 T_0 을 포함하여 라우터의 비밀키 SK 로 서명 되어 모든 인접 라우터로 분배된다.

$$ALSU : [\ \text{라우터 ID}, T_0, H^t(r_1), G^t(r_1), \ldots, H^t(r_j), G^t(r_j), \ldots, H^t(r_l), G^t(r_l)\ ]$$

인접 라우터로 부터 ALSU 를 받는 수신 라우터는 서명을 확인하여 T_0 가 새로운 것이면, 전체 ALSU 를 저장하고, 그 후에 갱신될 때 송신 라우터는 다음의 과정으로 $CLSU_i$ (i 는 session index)를 생성하여 전달하게 된다.

[CLSU 생성]

(1) 모든 링크 L_j ($1 \le j \le l$) 와 $CLSU_i$ ($1 \le i < t$) 마다 Link State Flag (LSF) 는 다음과 같이 정의된다.
$$LSF_i = [\ LF_i(1), \ldots, LF_i(l)\ ]$$
여기서, $LF_i(j)$는 L_j 가 UP 상태이면 1 이고, DOWN 상태이면 0 이다.

(2) 모든 링크 L_j ($1 \le j \le l$) 와 $CLSU_i$ ($1 \le i < t$) 마다 Link State Vector (LSV)는 다음과 같이 정의된다.
$$LSV_i = [\ LS_i(1), \ldots, LS_i(l)\ ]$$
여기서, $LS_i(j)$는 $LF_i(j)$가 1 이면 $H^{t-i}(r_j)$ 이고, $LF_i(j)$ 가 0 이면 $G^{t-i}(r_j)$ 이다.

(3) 이 과정의 결과로 다음과 같이 $CLSU_i$ 가 생성된다.
$$CLSU_i = [\ \text{라우터 ID}, i, T_i, LSF_i, LSV_i\ ]$$

이 후에 $CLSU_i$ 를 받은 수신 라우터는 다음과 같은 방법으로 $CLSU_i$ 를 처리한다.

[CLSU 처리]

(1) 현재 엔트리에서 라우터 ID 를 찾는다.

(2) $i > p$ 와 $T_i > T_p$ 를 확인한다. 이때, T_p 는 가장 최근에 저장된 timestamp 를 지칭한다.($CLSU_i$ 에 포함). 대부분의 경우, $p = i-1$. 만약, $i-p > 1$ 이면 인접한 라우터가 $i-p-1$ 개의 CLSU 를 이전에 수신하지 못한 경우이다.

(3) $CLSU_i$ 에 반영되어 있는 각각의 링크 L_j ($0<j<l$)의 상태가 변경되지 않았으면, 즉 $LF_i(j)=LF_p(j)$이면, 다음을 계산하고
$$LF_i(j) = 0 \text{ 이면 } G^{i-p}(LS_i(j))$$
$$LF_i(j) = 1 \text{ 이면 } H^{i-p}(LS_i(j))$$
$LS_p(j)$와 비교하여, 일치하지 않으면 폐기한다. 반면에 L_j 의 상태가 변경 되었으면, 즉 $LF_i(j) \neq$

$LF_p(j)$이면, 다음을 계산하고

$LF_i(j) = 0$ 이면 $G^i(G^{t-i}(R_i))$

$LF_i(j) = 1$ 이면 $H^i(H^{n-i}(R_i))$

$LS_i(j)$와 비교한 후, 역시 일치하지 않으면 폐기한다.

(4) 전체 LSV_i가 인증된 후, LSV_p는 LSV_i로 교체된다.

3. 확장된 FLS 기법의 제안

2 장에서 기술하였던 FLS 기법의 문제점은 단지 두 가지의 링크-상태, 즉 UP 과 DOWN 상태 만을 기술할 수 있다는 점이다. 이를 위해서는 2 개의 해쉬체인이 필요하게 된다. 이번 장에서는 링크-상태가 2 개 이상의 다중 값을 가질 경우에 다중 체인을 유지 관리함으로써 발생되는 메모리 복잡도의 증대를 해결하기 위한 새로운 확장된 FLS 기법을 소개한다. 이를 위해 다음 절에서 이중 해쉬체인(dual hash chain) 이라는 개념을 새로이 제안한다.

3.1 이중 해쉬체인(dual hash chain)

2 개의 seed 값 x 와 y 를 임의로 선정하고, 일방향 해쉬함수 H 를 이용하여 다음과 같이 길이가 n 인 2 개의 해쉬체인을 생성한다.

$$H^1(x), H^2(x), \ldots, H^k(x), \ldots, H^n(x)$$
$$H^n(y), H^{n-1}(y), \ldots, H^{n-k}(y), \ldots, H^1(y)$$

위에서 $H^n(x) \oplus H^n(y) = v$ 을 확인자(verifier)로 정의한다. 이 해쉬체인을 작성한 사용자는 확인자 v 를 디지털 서명한 값을 다른 사용자에게 보내고, 서명을 수신한 사용자는 그 서명을 작성자의 공개키를 이용하여 확인한 후에, 그 값을 저장한다. 차후에, 그 해쉬체인의 작성자는 임의의 k (= 1, 2, ..., n-1) 값을 다음과 같이 무결성이 보장되게 보낼 수가 있게 된다.

$$<n\text{-}k, H^k(x)>, <k, H^{n-k}(y)>$$

이 메시지의 수신자는 $H^k(x)$와 $H^{n-k}(y)$에, 각각 n-k 번과 k 번의 해쉬함수를 반복 적용하여 나온 결과값 $H^n(x)$와 $H^n(y)$을 Ex-OR 한 값과, 자신이 저장하고 있던 확인자 v 와 일치하는지를 확인하게 된다.

여기에서 하나가 아닌 두 개의 해쉬체인을 생성하는 이유는 하나의 해쉬체인을 사용하는 경우에는 무결성을 위협하는 공격이 가해질 수 있기 때문이다. 즉, 하나의 해쉬체인을 사용할 경우, 메시지 $<n\text{-}k, H^k(x)>$를 전송하게 되면, $H^{k+1}(x)$ 이 $H^k(x)$ 로부터 쉽게 계산되어질 수가 있기 때문에 공격자는 $<n\text{-}k, H^k(x)>$ 을 $<n\text{-}k\text{-}1, H^{k+1}(x)>$ 로 변조하여 보냄으로써 k 대신 k+1 의 값을 보낸 효과를 얻을 수가 있게 된다. 따라서, 무결성이 보장되지 않는다.

본 절에서 제시한 이중 해쉬체인(dual hash chain)의 경우에 공격자가 k 가 아닌 k+1 의 값으로 변조하기 위해서는 $<k, H^{n-k}(y)>$ 역시 $<k+1, H^{n-k-1}(y)>$으로 변경을 시켜야 하지만, 이는 일방향 해쉬함수의 특성상 불가능한 작업이 된다.

3.2 이중 해쉬체인에 기반하는 LSU 의 생성 및 처리

이전 절에서 기술한 이중 해쉬체인을 이용하면 기존 FLS 기법의 문제점을 해결할 수가 있게 된다. 즉, 두 개 이상의 다중 링크-상태 값을 전달할 수가 있다.

[ALSU 생성 및 처리]

각각의 라우터는 인접하는 링크마다 임의의 seed 값 x_1 과 y_1 그리고 해쉬함수 H 를 이용하여 길이가 n 인 두 개의 해쉬체인을 생성하고, 해쉬체인의 길이 n 은 라우터와 인접한 링크가 가질 수 있는 링크-상태 값 k 의 개수를 고려하여 설정한다. 즉, k = 1, 2, ..., n-1 값을 가질 수 있으며, 여기에서 k-1 은 링크의 DOWN 을 의미하며, 1 부터 k-2 까지의 값이 UP 상태에 있는 링크의 다양한 상태를 표시하게 된다.

또한, 라우터가 인접한 라우터에 보내는 링크-상태 값은 매 세션 i 마다 계속되어지기 때문에, 다음과 같은 세션 해쉬체인(session hash chain)을 정의한다. $H_0 = IV$ 를 초기화 벡터, $x_{i+1} = x_i + 1$ 와 $y_{i+1} = y_i + 1$ 을 매 세션마다 이중 해쉬체인에 사용될 seed 값이라고 하자. 이때, i = 1, 2, ..., t 이고 t 는 정의되는 세션 해쉬체인의 최대 길이가 된다.

$$H(H_0 \oplus v_1) = H_1$$
$$H(H_1 \oplus v_2) = H_2$$
$$H(H_2 \oplus v_3) = H_3$$
$$\vdots$$
$$H(H_{t-3} \oplus v_{t-2}) = H_{t-2}$$
$$H(H_{t-2} \oplus v_{t-1}) = H_{t-1}$$
$$H(H_{t-1} \oplus v_t) = H_t$$

위에서 $v_{t-i+1} = H^n(x_i) \oplus H^n(y_i)$, $(1 \le i \le t)$ 는 이중 해쉬체인에서의 확인자를 의미한다. 라우터는 인접한 라우터에 먼저 H_t 와 H_{t-1} 을 디지털 서명한 값을 보내고, 해당 라우터는 서명한 값을 확인한 후에 그 값을 저장하게 된다. 이는 결국 ALSU 를 생성하여 처리하는 과정이 되는데, 좀 더 자세히 ALSU 의 메시지를 기술하면 다음과 같다.

$$ALSU : [\ \text{라우터 ID}, H_t, H_{t-1}, T_0\ ]_{SK}$$

인접 라우터로 부터 ALSU 를 받은 라우터는 서명을 확인하고, T_0 이 새로운 것이면 전체 ALSU 를 저장한다.

[CLSU 생성 및 처리]

CLSU 를 전달하고자 하는 라우터는 다음과 같이 $CLSU_i$(i 는 session index)를 생성하여 전달하게 된다.

$$CLSU_i : [\ \text{라우터 ID}, i, T_i, H_{t-i-1}, n\text{-}k_i, H^{k_i}(x_i), k_i, H^{n-k_i}(y_i)\ ]$$

위에서, k_i (= 1, 2, ..., n-1)는 i 번째 세션에서의 링크-상

태 값을 지칭한다.

CLSU 를 수신한 라우터는 CLSU 가 정당한 라우터로부터 전송된 메시지임을 확인하기 위해 우선 $CLSU_i$ 의 T_i 를 검사하여 T_i 가 새로운 것이면 $CLSU_i$ 에 포함되어 있는 $H^{k_i}(x_i)$과 $H^{n-k_i}(y_i)$ 값에 각각 $n-k_i$ 번과 k_i 번의 해쉬함수를 반복 적용하여 나온 결과값 $H^n(x_i)$와 $H^n(y_i)$를 이용하여 확인자(verifier) v_{t-i+1} 를 구성하고, 이전 세션에서 사용되었던 H_{t-i} 를 이용하여 $H(H_{t-i} \oplus v_{t-i+1})$ 값이 H_{t-i+1} 과 일치하는지 확인하고, 성공적으로 확인이 되면 라우터는 현재 저장되어 있는 $CLSU_{i-1}$ 대신 $CLSU_i$ 를 저장한다.

여기에서 첫번째 CLSU 인 $CLSU_1$ 일 경우에는 이전 CLSU 가 없으므로, ALSU 에 포함되어 있는 세션 해쉬체인 값인 H_t 와 H_{t-1} 을 사용하여 계산한다. 또 t 번째 CLSU 인 $CLSU_t$ 일 경우는 마지막 단계이므로 v_1 을 계산할 수 있는 값만을 전송하게 된다. 다음 세션부터는 새로운 ASLU 의 작성과 전송을 통하여 생성된 새로운 세션 해쉬체인을 사용하게 된다.

위에서 $CLSU_i$ 를 전송함에 있어서 H_{t-i} 가 아닌 다음 세션에서 사용하는 H_{t-i-1} 을 전송하는 이유는 $CLSU_i$ 에 H_{t-i} 를 전송하게 되면, 공격자는 $H_{t-i} \oplus v_{t-i+1} = H'_{t-i} \oplus v'_{t-i+1}$ 조건을 만족하는 v'_{t-i+1} 에 해당하는 이중 해쉬체인에서의 해당 값, 그리고 H'_{t-i} 값을 선정하여 $H(H'_{t-i} \oplus v'_{t-i+1})$ 을 계산할 수 있고 이는 결국 H_{t-i+1} 값과 일치하게 됨으로써 정당한 라우터 임을 가장하게 된다. 이런 이유로 $CLSU_i$ 에 다음 세션에 사용될 H_{t-i-1} 을 전송함으로써 공격으로부터 보호한다.

4. 기존방식과 제안방식의 비교

기존의 FLS 는 각 링크 당, 단지 UP 과 DOWN 의 2 가지 상태만을 표현하기 위해 제안되었다. 하지만, 이는 다양한 링크의 상태를 가정하는 링크-상태 라우팅 프로토콜에는 적절하지 못한 대안이다. 만약, 이 방식을 그대로 n (> 2)개 이상의 상태를 나타내는 방식으로 전환할 경우에는 각 링크마다 n 개의 독립적인 해쉬체인이 필요하게 된다. 이 경우에는 두 가지의 구현방안이 있는데, 첫째는 n 개의 초기값 만을 저장하고, 매 세션마다 해당 해쉬 값을 계산 해 주는 경우이고, 두 번째 방식은 길이가 t 인 n 개의 전체 해쉬체인을 저장하여 사용하는 것이다. 하지만, 구현 측면에서는 두 가지 방안 모두 메모리 복잡도나 계산 복잡도 측면을 고려할 때 바람직하지 못하다.

예를 들어, 링크-상태 값은 $n = 20$, 그리고 라우팅 정보는 매 1 분마다 교환되고, ALSU 는 하루에 한번 전달된다고 가정한다면 대략 $t = 1000$ 정도로 추정이 가능하다. 128 비트의 해쉬 값의 경우에 첫번째 방안은 $128 \times 20 \times 2$ 비트의 메모리와 매 세션마다 $1000 - i$ 번($i = 1, 2, \ldots$)의 해쉬 계산이 소요된다. 두번째의 경우에는 모든 해쉬 값을 저장하고 사용하는 경우이기 때문에 각 링크마다 $128 \times 1000 \times 20$ 비트의 메모리가 소요된다.

본 논문에서 제안하는 방식은 이 두 가지 방안에 대한 절충안으로 볼 수가 있다. 즉, 세션 해쉬체인의 t 개의 해쉬 값은 저장하고, 매 세션마다, 링크-상태 값은 이중 해쉬체인을 통해서 계산하는 것이다. 이 경우에는 128×1000 비트의 메모리가 소요되고, 그리고 링크-상태 값을 위해서 매 세션마다 $20 = (20-k_i) + k_i$ 번의 해쉬 계산이 요구된다.

5. 결론

본 논문에서는 링크-상태 라우팅의 라우팅 비용을 줄이는 방법에 있어서 기존에 제안되었던 방법을 설명하고, 기존에 제안되었던 방법을 보완하여 링크-상태가 자주 변동하는 환경에서 효과적이면서, 링크-상태가 다중값을 가지는 경우에 효율적으로 라우팅 비용을 줄일 수 있는 방법에 대해 기술했다.

참고문헌

[1] J. Moy, "OSPF Version 2", RFC 1583, 1994.

[2] R. Hauser, T. Przygienda, and G..Tsudik, "Reducing the Cost of Security in Link State Routing", Computer Networks and ISDN Systems, Vol. 31 (8), Page 885-894, 1999.

[3] R. Perlman, "Network Layer Protocols with Byzantine Robustness", Ph.D. Thesis, Department of Electrical Engineering and Computer Science, MIT, Aug. 1988.

[4] Michael T. Goodrich, "Efficient and Secure Network Routing Algorithms", provisional patent filing, U.S.A, 2001.

[5] K. Zhang, "Efficient Protocols for Signing Routing Messages", In Symposium on Network and Distributed Systems Security (NDSS '98), San Diego, California, 1998.

[6] S. Cheung, An Efficient Message Authentication Scheme for Link State Routing", Proceedings of the 13th Annual Computer Security Applications Conference, San Diego, California, Page 90-98, 1997.

[7] G. Malkin, "RIP Version 2", RFC 2453, 1998.