논문 17-42-05-09 The Journal of Korean Institute of Communications and Information Sciences '17-05 Vol.42 No.05
https://doi.org/10.7840/kics.2017.42.5.999

무선 네트워크에서 파이프라인 네트워크 코딩 기반 메시지 및 노드 인증

안 명 기◆, 조 영 종°, 강 경 란*

PNC(Pipeline Network Coding)-Based Message and Node Authentication in Wireless Networks

Myeong-Gi Ahn◆, Young-Jong Cho°, Kyungran Kang*

요 약

본 논문에서는 무선 네트워크 환경에서 효율적인 데이터 전달을 위한 파이프라인 네트워크 코딩(Pipeline Network Coding) 기법과 데이터의 무결성을 검증하기 위한 데이터 인증 기법, 가상 송신자에 대한 노드 인증 기법을 제안한다. 파이프라인 네트워크 코딩 기법은 네트워크 코딩을 수행하는 중계 노드가 송신자 대신 데이터를 전달함으로써 전체적인 네트워크 성능을 향상시키는 기법이다. 그러나 네트워크 코딩은 악의적인 공격자가 데이터를 위·변조하여 네트워크에 주입하는 공격인 오염 공격(pollution attack)에 취약하다. 이를 방어하기 위해 HMAC(Hash-based Message Authentication Code)을 사용한다. 이때 데이터 인증에 사용되는 태그를 생성하기 위해서는 인증을 수행하는 노드들에게 키를 분배해야한다. 키 분배에 따른 오버헤드를 최소화하기 위해 해쉬 체인을 적용하였다. 가상 송신자에 대한 인증 기법으로는 null 벡터를 사용한다. 최종적으로 제안 기법에 대한 안전성과 복잡도를 분석하고, 시뮬레이션을 통해 성능을 분석하였다.

Key Words : Network Coding, Wireless Network, HMAC, Null-vector, Authentication

ABSTRACT

In this paper, we propose a pipeline network coding (PNC) scheme for efficient data transmission in wireless networks, a data authentication scheme for verifying the integrity of data, and a node authentication scheme for a virtual source. PNC is a technique that improves the overall network performance by relaying data such that the relay node performing network coding transmits to the sender instead. However, network coding is vulnerable to a pollution attack, which is an attack by a malicious attacker to inject modified data into the network. To prevent this, hash-based message authentication code (HMAC) is used. For this purpose, in order to generate a tag used for data authentication, a key must be distributed to the nodes performing authentication. We applied a hash chain to minimize the overhead of key distribution. A null vector is used as the authentication scheme for the virtual source. Finally, we analyze the safety and complexity of the proposed scheme and show he performance through simulation.

◆ First Author : LIG Nex1 Co., Ltd., myeonggi.ahn@lignex1.com, 정회원
° Corresponding Author : Ajou University Department of Software and Computer Engineering, yjcho@ajou.ac.kr, 종신회원
* Ajou University Department of Software and Computer Engineering, korykang@ajou.ac.kr, 정회원
논문번호 : KICS2017-02-034, Received February 2, 2017; Revised April 18, 2017; Accepted May 17, 2017

Ⅰ. 서 론

네트워크의 빠른 발달로 인하여 유·무선 네트워크 트래픽이 급증하는 문제점이 발생하고 있다. 이를 해결하기 위해 네트워크 코딩(network coding) 기법이 제안되었다[1]. 네트워크 코딩은 중계 노드가 수신한 메시지를 인코딩(encoding)하여 전달함으로써 전체 네트워크 성능을 향상시키는 기법이다. 네트워크 코딩 기법 중에서 특정 크기의 유한 체(finite field)에서 무작위로 계수들을 선택하여 패킷과 결합하는 기법인 RLNC(Random Linear Network Coding)가 가장 많이 사용되고 있다[2]. 특히 네트워크 코딩은 무선 네트워크에서 에너지 효율, 지연시간, 신뢰성 등의 성능향상에 있어서 다양한 연구를 통해 증명되었다[3-6].

최근에는 이러한 네트워크 코딩의 이점을 극대화하기 위하여 수신자가 가상 소스가 되어 실제 송신자가 생성하는 네트워크 코딩 패킷을 대신 전달하고, 이를 파이프라인 형식으로 전달함으로써 에너지 효율과 네트워크 처리율을 향상시키는 기법이 제안되었다[7].

하지만 가상 소스가 실제 송신자에게 네트워크 코딩 패킷 전달 권한을 요청할 때, 악의적인 공격자가 자신이 적합한 노드인 것처럼 위장하여 권한요청을 함으로써 네트워크 코딩 패킷 전달 권한을 얻을 수 있다. 이와 같은 취약점을 악용하여 공격자는 전체 네트워크 성능을 저하시키는 공격을 가할 수 있다. 이러한 문제점을 해결하기 위해 가상 소스의 권한요청이 적합한 노드로부터 온 것인지 확인하는 인증절차가 필요하다. 또한, 네트워크 코딩이 적용되는 환경은 위·변조된 데이터를 네트워크에 주입시키는 오염 공격(pollution attack)에 매우 취약하다. 이를 위해 위·변조된 메시지를 탐지 및 폐기하기 위한 인증 절차가 필요하다.

본 논문에서는 기존 논문에서처럼 수신자가 가상 소스 역할을 하는 것이 아니라 중계 노드가 가상 소스 역할을 함으로써 데이터의 컨트롤 루프(control loop)를 감소시켜 전체 네트워크 성능을 향상시키는 기법을 제안한다. 그리고 네트워크 코딩 환경에서 발생할 수 있는 오염 공격을 방어하기 위해 HMAC 기반의 인증 기법을 사용한다. 이때 인증에 필요한 키 분배 시에 발생하는 오버헤드를 최소화하기 위해 해쉬 체인을 적용하여 키를 분배하는 메시지 인증 기법을 제안한다. 또한, 적합한 중계 노드가 네트워크 코딩 패킷에 대한 전달 권한을 요청할 때 null 벡터를 이용한 인증 기법을 제안한다. 최종적으로 제안한 기법의 효율성을 검증하기 위해 시뮬레이션을 통해 기존 논문들과 성능을 비교·분석하고, 수학적 모델을 사용하여 공격에 대한 안전성을 검증한다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구들을 알아보고, 3장에서는 제안 기법에 대한 기술적 고려사항 등을 설명한다. 4장 에서는 논문에서 제안한 기법들에 대해서 기술하고, 5장에서는 이에 따른 안전성 및 복잡도를 분석한다. 6장에서는 시뮬레이션 결과를 통한 성능 분석이 기술되어 있다. 7장에서는 제안 기법에 대한 성과로 결론을 맺는다.

Ⅱ. 관련 연구

2.1 CodePipe

CodePipe는 무선 네트워크에 다수의 수신자들에게 멀티캐스트로 데이터를 전달하는 환경을 가정하고 있다. 이때 네트워크 코딩을 사용하여 에너지 효율을 높이는 것을 목적으로 한다. 최종 수신자들은 디코딩이 가능한 만큼의 패킷을 수신하였을 때 실제 송신자 대신 네트워크 코딩 패킷을 대신 전달한다. 이로 인해 네트워크에 존재하는 모든 노드들이 전송해야하는 패킷의 수를 감소시켜 에너지 효율을 향상시킬 수 있었다.

기존 기회적 라우팅 프로토콜인 MORE나 Pacifier 등은 패킷을 수신 받는 노드들의 공정성을 고려하지 않았다[8,9]. 이는 수신 받아야할 현재 패킷을 모두 받은 노드들은 패킷을 받지 못한 다른 노드들이 패킷을

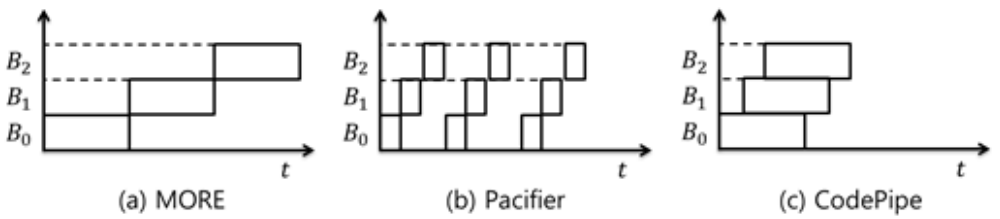

그림 1. 기회적 라우팅 프로토콜의 패킷 전달 방법 비교
Fig. 1. Comparison of packet delivery methods of opportunistic routing protocols

모두 받을 때까지 기다려야한다는 단점이 있다. 또한, 모든 노드들이 패킷을 모두 수신할 때까지 송신자는 계속해서 패킷을 전송해야하므로 낮은 에너지 효율성을 보인다. CodePipe는 네트워크 코딩 패킷을 디코딩이 가능한 만큼의 메시지를 수신한 노드가 송신자 대신 현재 배치의 메시지를 대신 전달함으로써 전체 패킷 전송 횟수를 최소화하여 에너지 효율성을 향상시켰다.

그림 2는 기회적 라우팅 프로토콜들의 배치에 전달 기법을 비교한 것이다. MORE에서 송신자는 모든 수신자가 현재 배치에 해당하는 패킷을 받을 때까지 지속적으로 패킷을 전달한다. 이는 수신이 완료된 수신자가 다음 패킷을 기다려야 하는 공정성 문제점이 있다. 이를 해결하기 위해 Pacifier에서는 송신자가 패킷을 전달할 때 라운드 로빈(round robin) 방식을 사용하여 공정성 문제를 해결하였다. 하지만, 전체 배치를 전달할 동안 모든 배치의 패킷을 지속적으로 전송해야 하는 문제점이 있다. CodePipe에서는 각 배치에 전송해야 하는 패킷을 파이프라인 형태로 전송하여 공정성문제와 과도한 패킷 전송량으로 인한 문제를 해결하였다.

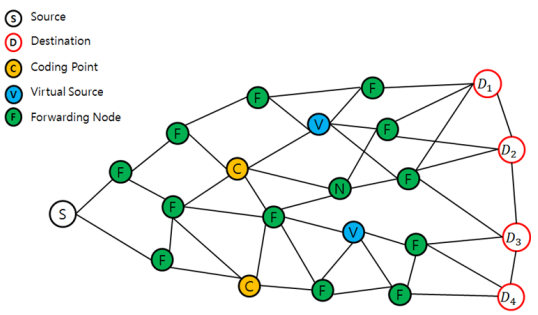

그림 2. 네트워크 모델
Fig. 2. Network model

2.2 메시지 인증 기법

메시지 인증은 송신자가 보낸 메시지가 위·변조 되었는지 확인하기 위한 절차이다. 특히 네트워크 코딩 환경에 취약한 오염 공격을 방어하기 위해서는 메시지 인증 절차는 필수적이다. 다양한 메시지 인증 기법이 있지만 네트워크 코딩 환경에서 사용하는 인증 기법으로는 null 벡터를 사용한 인증 기법과 MAC(Message Authentication Code)을 사용한 인증 기법이 존재한다.

Null 벡터를 사용한 인증 기법은 네트워크 코딩 패킷에 대한 null space를 생성하여 인증을 수행한다[10-11]. 수신한 네트워크 코딩 패킷과 null space의 내적 값이 0이 되면 메시지가 위·변조 되지 않았다는 것을 검증한다. 선형대수학에서 null space는 식 (1)에서처럼 $m \times n$ 행렬 A에 대하여 $Ax = 0$을 만족하는 벡터 x의 집합을 의미하며 $Null(A)$로 표현한다.

$$Null(A) = \{x : x \in R^n,\ Ax = 0\} \tag{1}$$

본문에서는 노드 인증 시 사용되는 null 벡터 인증에 null space 개념을 사용한다. 패킷 p_i는 N차원을 갖는 벡터라고 가정하고, 이러한 패킷 p_i가 N개만큼 있을 때 이는 식 (2)처럼 $N \times N$ 행렬로 표현이 가능하다.

$$P = \begin{bmatrix} p_{1,1} \cdots p_{1,N} \\ \vdots \ \ddots \ \vdots \\ p_{N,1} \cdots p_{N,N} \end{bmatrix} \tag{2}$$

이때 행렬 P에 대한 인증 방법은 P의 null space 중 하나의 vector를 선택하여 원본 패킷과 내적(dot product) 연산을 수행하는 것이다. 내적 연산 결과 값이 0이 나왔을 때 인증 절차를 통과하게 된다.

MAC은 메시지에 대한 무결성을 검증하기 위한 tag를 생성하여 송신자와 수신자 간에 교환된 메시지의 위·변조 여부를 탐지하기 위한 인증 기법이다. 네트워크 코딩 환경에서 사용하는 대표적인 MAC 기법으로는 HMAC(Hash-based Message Authentication Code)이 있다. 네트워크 코딩 환경에서는 선형 결합 시 tag도 선형 결합이 적용된다. 이때 tag 검증을 위해서는 tag의 무결성이 보장되어야 한다. 이를 위해 HHF(Homomorphism Hash function)을 사용한다[12,13]. HHF를 tag에 적용하게 되면 식 (3)처럼 결과 값에 임의의 계수를 곱해도 원래의 tag 값에 영향이 없다. ($x : key, c$: 상수)

$$c \bullet HHF(x, tag) = HHF(x, c \bullet tag) \tag{3}$$

기존 MAC에 비해서 HMAC은 tag 생성 시 필요한 키를 해쉬 함수를 사용한다. 해쉬 함수를 사용하여 생성된 키 값은 항상 일정한 값을 유지하기 때문에 키 관리에 있어서 유리하다.

2.3 키 분배 기법

네트워크 코딩 패킷 인증을 위해서 송신자는 인증에 필요한 키를 인증을 수행하는 노드들에게 분배해

야한다. 키 분배 방식으로는 대칭키 방식, 비대칭키 방식이 있다. 대칭키 방식은 송신자와 수신자가 동일한 키를 사용하여 인증을 수행하는 방식이고, 비대칭키 방식은 다른 종류의 키를 사용하여 인증을 수행하는 방식이다. 이때 비대칭키 방식을 사용하여 비밀키와 공개키 간의 관례를 수식화하여 키를 분배하는 기법이 제안되기도 했다[14].

키 분배 시에 문제점으로는 키 분배 오버헤드와 키 저장 공간에 대한 오버헤드가 있다. 이를 해결하기 위해 해쉬 체인 기법을 적용하여 키를 분배하는 기법이 제안되었다[12]. 하지만 이 방식은 키 노출에 의한 인증 지연시간 때문에 모든 노드들이 실시간으로 메시지 인증을 수행하지 못한다는 단점이 있다.

2.4 Hash chain

해쉬 체인은 임의의 $seed$를 입력으로 하여 연속해서 해쉬 값을 생성하는 기법이다[15]. 해쉬 체인의 해쉬 값을 계산하기 위해서는 $seed$와 반복 횟수인 n이 주어져야 한다. 임의의 $seed$ x와 n에 대해 해쉬 체인을 적용하면 식 (4)와 같다.

$$c_n = h(x), c_{n-1} = h(h(x)), \ldots, c_1 = h^n(x), \\ c_0 = h^{n+1}(x) \quad (4)$$

위 식에서 $c_{n-i+1} = h^i(x)$ 값은 x를 i번 해쉬 함수를 적용한 값을 의미하며, $c_0 = h^{n+1}(x)$는 해쉬 함수를 통해 얻는 마지막 값을 의미한다.

해쉬 체인은 해쉬 함수의 성질을 그대로 포함하고 있기 때문에 일방향성을 갖고 있다.

Ⅲ. 제안 기법

3.1 네트워크 모델

본 논문에서 고려하는 네트워크 모델은 그림 2처럼 네트워크 코딩을 적용한 무선 네트워크 환경을 고려하고 있다. 각 노드들에 대한 역할은 다음과 같다.

Source S는 N개의 데이터 블록을 다수의 목적지 노드로 전달한다. 각 데이터 블록은 N차원의 패킷으로 구성되어 있다. S는 패킷에 네트워크 코딩을 적용하여 다음 노드들에게 전달한다. 이때 악의적인 공격자로 인한 오염 공격을 방어하기 위해 HMAC을 사용하여 생성된 tag를 추가하여 전송한다.

Coding point C는 네트워크 코딩을 수행하는 노드로써 네트워크 코딩 패킷을 수신하면 위·변조 여부를 탐지하기 위한 인증을 수행하고, RLNC를 적용하여 다음 노드에게 전달한다.

Virtual source V는 C의 역할을 동일하게 수행함과 동시에 S에게 네트워크 코딩 패킷 전달 권한을 요청한다. V는 디코딩이 가능한 만큼의 메시지를 수신하면 원본 메시지를 복구한다. 그리고 V가 적합한 노드인지 확인하기 위해 null 벡터를 생성하고 S에게 전달하여 인증을 수행한다.

Destination D는 수신한 메시지에 대한 인증 및 디코딩을 수행한다.

Forwarding node F는 네트워크 코딩을 수행하지 못하는 노드로써 오직 메시지 전달만을 수행한다.

3.2 효율적인 데이터 전달 기법

CodePipe에서처럼 최종 수신자가 가상 소스의 역할을 하는 것이 아니라 네트워크 코딩을 수행하는 중계 노드들이 가상 소스 역할을 수행하여 효율적으로 데이터를 전달하는 기법을 제안한다. 그림 3은 제안 기법의 전체적인 데이터 전달 절차를 나타낸 그림이다. 이 과정에서 수행되는 데이터 인증 및 노드 인증은 다음 절에서 설명한다.

1) S는 패킷에 임의의 계수를 곱한 후 선형결합을 적용하여 네트워크 코딩 패킷을 생성한다. 또한, 오염 공격을 방어하기 위해 tag를 생성하여 전달한다.
2) C는 수신한 데이터에 대해 적합한 노드로부터 전송된 데이터인지 확인하기 위한 데이터 인증 절차를 수행한다. 일정 시간동안 네트워크 코딩

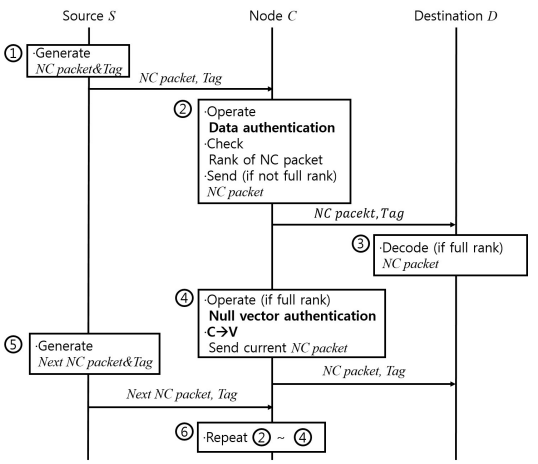

그림 3. 효율적인 데이터 전달 절차
Fig. 3. Efficient data transmission procedure

버퍼에 네트워크 코딩 패킷을 저장하면서 해당 패킷의 rank를 체크한다. 버퍼에 저장되어 있는 패킷의 rank가 full rank가 되지 않았으면 RLNC를 적용하여 다음 노드에게 전송한다.

3) D는 수신한 네트워크 코딩 패킷을 버퍼에 저장한다. 패킷의 rank를 체크하여 full rank가 되었을 때 디코딩을 수행하여 원본 데이터를 복구한다.
4) C의 버퍼에 저장되어 있는 네트워크 코딩 패킷이 full rank가 되면 null 벡터 인증을 수행한다. Null 벡터 인증이 완료되면 해당 C는 V로 지정되고, S대신 현재 네트워크 코딩 패킷을 다음 노드들에게 전달한다.
5) S는 다음 generation에 대한 네트워크 코딩 패킷과 태그를 생성하여 노드들에게 전달한다.
6) D가 모든 패킷을 수신할 때까지 ②~⑤ 과정을 반복한다.

3.3 Data authentication

네트워크 코딩이 적용되는 환경은 오염 공격에 취약하다. 이를 방어하기 위해 데이터 인증 기법은 필수적이다. 데이터 인증 기법을 사용할 때에는 인증에 필요한 키를 효율적으로 분배해야 한다. 본 논문에서는 해쉬 체인 기반의 키 분배 기법을 사용한 데이터 인증 기법을 제안하여 전체적인 절차는 그림 4와 같다. 여기서 KDC는 인증에 필요한 $seed$를 분배하는 키 분배 센터이다. 표 1은 데이터 인증 기법에서 사용되는 표기들에 대한 의미를 나타낸다.

1) KDC는 tag 생성에 필요한 임의의 3개의 $seed$를 생성하여 S에게 전달한다.
2) S는 KDC로부터 받은 $seed$들을 사용하여 해쉬

표 1. 표기법
Table 1. Notations

Notation	Definition
N	Generation size
$h(x)$	Priveate key
$h(y), h(z)$	Public keyt
Tag_M	Tag created using HMAC
M	Network coding packet
$h(f)$	Fake private key
Tag_{M_f}	Fake tag
M_f	Fake network coding packet
KDC	Key Distribution Center
NC	Network Coding

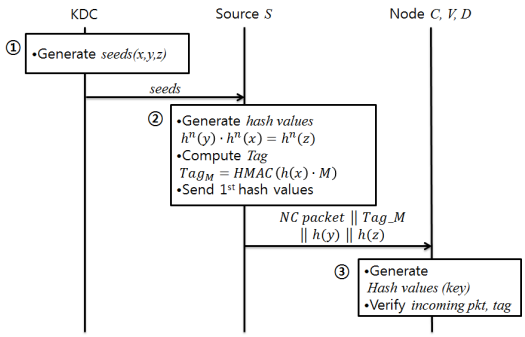

그림 4. 데이터 인증 절차
Fig. 4. Data authentication procedure

값을 생성한다. 각 해쉬 값들은 식 (5)와 같은 관계가 성립한다.

$$h^N(y) \bullet h^N(x) = h^N(z) \tag{5}$$

S는 $h(x)$를 비밀키로 하여 네트워크 코딩 패킷에 대한 tag를 식 (6)과 같이 생성한다.

$$Tag_M = HMAC(h(x), M) \tag{6}$$

네트워크 코딩 패킷, tag, 공개키로 사용되는 $h(y), h(z)$를 메시지 인증을 수행하는 노드들에게 전달한다. 기존 논문에서처럼 모든 N 만큼의 키를 전달하는 것이 아니라 첫 번째 키에 해당하는 해쉬 값만 전달함으로써 키 분배에 따른 오버헤드와 키 저장 공간 오버헤드를 줄일 수 있다.

3) 메시지 인증을 수행하는 각 노드들은 수신한 tag, 네트워크 코딩 패킷, $h(y), h(z)$를 사용하여 tag에 대한 검증을 수행한다. 식 (7)을 통해 좌변과 우변이 일치하면 메시지가 위·변조 되지 않았음을 확인한다. 2번째 메시지부터는 각 노드들이 직접 해쉬 함수를 적용하여 해당 해쉬 값을 생성하여 인증 절차를 수행한다.

$$\begin{aligned} h^N(z) \bullet M_N &= h^N(y) \bullet h^N(x) \bullet M_N \\ &= h^N(y) \bullet Tag_{M_N} \end{aligned} \tag{7}$$

3.4 Null 벡터 기반 노드 인증

효율적인 데이터 전송 기법에서 노드 V는 S에게 네트워크 코딩 패킷 전달 권한을 요청한다. 이때 악의적인 공격자가 V로 위장하여 S에게 네트워크 코딩 패킷 전달 권한을 요청할 수 있다. 이러한 취약점을 방

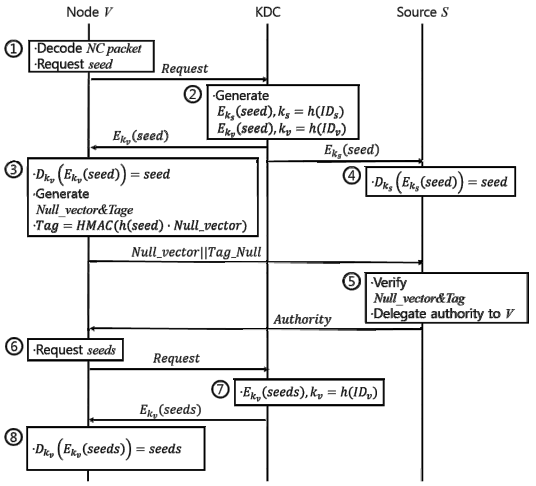

그림 5. Null 벡터 기반 노드 인증 절차
Fig. 5. Null-vector based node authentication procedure

어하기 위해 null 벡터 인증 절차를 제안하며 그림 5와 같다.

1) V는 디코딩이 가능한 만큼의 네트워크 코딩 패킷을 받으면 가우시안 소거법을 사용하여 원본 패킷을 구한다. 이때 null 벡터에 대한 무결성 검증을 위해 tag를 생성해야한다. 이를 위해 KDC에게 tag 생성에 필요한 $seed$를 요청한다.
2) KDC는 null 벡터에 대한 tag 생성을 위한 새로운 $seed$를 생성하고, S와 V의 ID를 키로 하여 암호화한 후 각각 S와 V에게 전달한다.
3) V는 자신의 ID를 키로 하여 수신한 $seed$를 복호화하고, 네트워크 코딩 패킷 전달 권한 요청을 위해 null 벡터와 tag를 생성하여 S에게 전달한다.
4) S는 자신의 ID를 키로 하여 수신한 $seed$를 복호화한다.
5) S는 수신한 null 벡터와 tag에 대한 검증이 완료되면, V에게 네트워크 코딩 패킷 전송 권한을 위임한다.
6) 네트워크 코딩 패킷 전송 권한을 위임 받은 V는 자신이 생성한 네트워크 코딩 패킷에 대한 tag를 생성하기 위해 KDC에게 $seed$를 요청한다.
7) KDC는 새로운 $seed$를 암호화한 후 V에게 전달한다.
8) V는 자신의 ID를 키로 하여 복호화한 후 새로운 $seed$를 구한다. V는 네트워크 코딩 패킷을 생성하고, 새로운 $seed$를 사용하여 데이터 인증에 필요한 tag를 생성하여 다음 노드들에게 전달한다.

Ⅳ. 시뮬레이션 결과 및 성능 분석

4.1 실험 환경

본 논문에서는 제안 기법의 성능을 분석하기 위해서 네트워크 시뮬레이터인 QualNet을 사용하였다. 표 2는 시뮬레이션을 위한 실험 환경이다.

표 2. 시뮬레이션 환경
Table 2. Simulation environments

Parameter	Value
Terrain size	1000m × 1000m
Number of nodes	30
Topology	Random
Receiving distance	250m
Carrier sensing range	550m
MAC protocol	802.11
Data rate	2Mbps
Packet size	1024 bytes
Interval	1 seconds
Hash function computation time	1.268(μs)

4.2 시뮬레이션 시나리오

시뮬레이션은 총 3가지의 시나리오로 구성하여 진행하였다.

첫 번째 시나리오는 효율적인 데이터 전달 기법의 성능을 분석하기 위한 시나리오이다. 하나의 소스 노드는 4개의 목적지 노드에게 데이터를 전달한다. 소스 노드와 목적지 노드 사이에는 네트워크 코딩을 수행하는 노드 C가 배치되어 있다. 실험은 토폴로지 상에서 C의 위치를 변경하면서 진행하였다. C의 위치 변경의 기준은 소스 노드로부터의 hop count를 기준으로 하였고, 2, 4, 6씩 변경하면서 실험을 측정하였다.

두 번째 시나리오는 데이터 인증에 대한 성능을 분석하였다. 공격자는 C에게 오염 공격을 시도한다. 시뮬레이션은 공격자의 비율을 변경하면서 진행하였다. 본 논문에서 해쉬 체인 기반의 데이터 인증 기법과 기존 논문에서 제시한 데이터 인증 기법인 KEPTE와 비교하여 성능을 분석하였다.

세 번째 시나리오는 null 벡터 인증에 대한 성능을 분석하였다. 공격자는 S에게 임의의 null 벡터를 생성하여 인증 절차를 통과하기 위한 공격을 시도한다. 시뮬레이션은 공격자의 비율을 변경하면서 진행하였다.

4.3 시뮬레이션 결과 및 성능 분석

앞서 설명한 3가지 시나리오에 대한 시뮬레이션 결과 그래프와 성능을 분석한다.

4.3.1 효율적인 데이터 전달 기법 성능 분석

그림 6은 코딩 포인트의 비율을 5%씩 증가시키면서 코딩 포인트의 위치에 따른 end-to-end delay를 나타내고 있다. 제안 기법의 가장 큰 장점은 코딩 포인트의 위치가 소스 노드와 근접할수록 컨트롤 루프의 길이가 짧아진다는 것이다. Hop count가 작을수록 컨트롤 루프의 길이가 짧다는 의미이다. 코딩 포인트의 위치가 소스 노드와 근접할수록 delay가 감소하는 것을 볼 수 있다. 또한, 코딩 포인트의 비율이 증가할수록 네트워크 코딩 이득이 증가하면서 delay가 감소하는 것을 볼 수 있다.

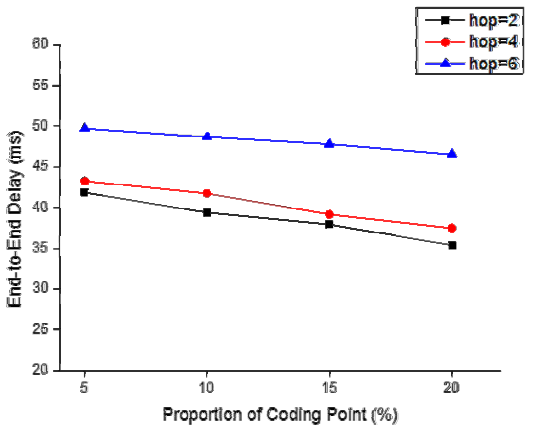

그림 6. 코딩 포인트 비율에 따른 hop count 별 end-to-end delay
Fig. 6. End-to-end delay as to the hop count according to the proportion of coding point

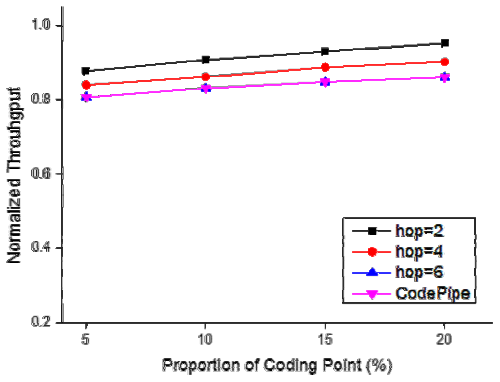

그림 7. 코딩 포인트 비율에 따른 hop count 별 normalized throughput
Fig. 7. Normalized throughput as to the hop count according to the proportion of coding point

그림 7은 코딩 포인트 비율을 5%씩 증가시키면서 코딩 포인트의 위치에 따른 normalized throughput을 나타내고 있다. CodePipe에서는 최종 수신자 노드 중에서 가상 소스가 선정되기 때문에 컨트롤 루프의 효과가 미비하다. 하지만 코딩 포인트의 위치를 변경하면서 실험한 결과를 보면 제안 기법의 강점을 볼 수 있다. 코딩 포인트의 비율이 증가할수록 네트워크 코딩 이득이 증가하게 되고, 코딩 포인트의 위치가 소스 노드와 가까울수록 컨트롤 루프의 길이가 짧아지기 때문에 측정한 normalized throughput이 증가하는 것을 볼 수 있다.

4.3.2 데이터 인증 성능 분석

그림 8은 데이터 인증 시 수행되는 키 분배 오버헤드를 N에 따라 측정한 그래프이다. 제안한 데이터 인증 기법과 KEPTE와 비교하여 실험을 진행하였다[14]. KEPTE는 모든 N에 대한 키를 전송하고, 제안 기법은 첫 번째에 해당하는 키만 전송한다. 이로 인해 제안 기법보다 KEPTE가 더 많은 키 분배 오버헤드를 보이고 있다. 제안 기법과 KEPTE의 키 분배 오버헤드가 N이 증가하여도 일정한 이유는 패킷에 키가 합쳐져서 전송되기 때문이다.

그림 9는 공격자 비율에 따른 전체 노드의 에너지 소비 비율을 나타내고 있다. N을 512와 1024로 변화시키면서 제안 기법과 KEPTE를 적용하였을 때 전체 노드의 에너지 소비 비율을 측정하였다. 에너지 소비 종류로는 데이터를 전송할 때 사용되는 에너지와 해쉬 함수를 수행할 때 발생되는 에너지가 있다. 공격자

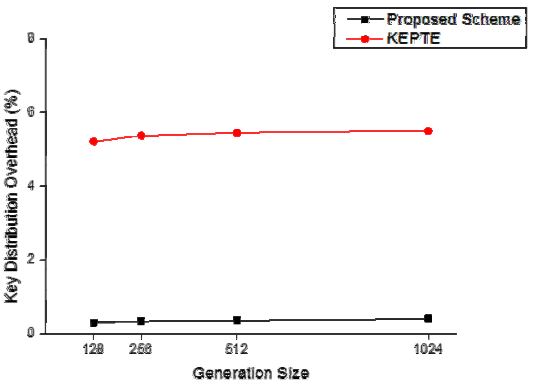

그림 8. Generation size에 따른 키 분배 오버헤드
Fig. 8. Key distribution overhead according to generation size

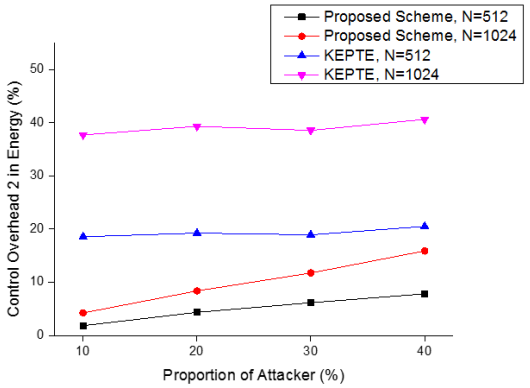

그림 9. 공격자 비율에 따른 에너지 오버헤드
Fig. 9. Control overhead 2 in energy according to the proportion of attacker

의 비율이 증가할수록 제안 기법의 에너지 소비 비율은 증가하긴 하지만 KEPTE의 에너지 소비 비율보다는 적은 값을 나타내고 있다. 이러한 이유는 해쉬 함수 계산 에너지보다 데이터 전송에 소비되는 에너지가 더 크기 때문이다. 즉 모든 노드들은 키를 전송하면서 에너지를 소비하기 때문에 제안 기법보다 KEPTE에서 더 많은 에너지를 소비하게 된다.

4.3.3 Null 벡터 인증 성능 분석

그림 10은 null 벡터 인증 수행 시 공격자 비율에 따른 normalized throughput을 나타내고 있다. 공격자가 null 벡터 공격을 수행하였을 때 제안 기법을 적용하여 공격을 방어하는 경우와 공격을 방어하지 않는 경우를 비교하였다. 공격을 방어하지 않았을 때는 normalized throughput이 현저하게 급감하는 것을 볼 수 있다. 즉 네트워크에 비정상적인 데이터가 유입되

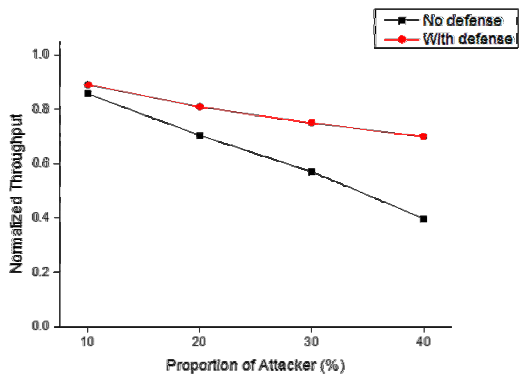

그림 10. 공격자 비율에 따른 normalized throughput
Fig. 10. Normalized throughput according to the proportion of attacker

었기 때문에 전체 네트워크 성능 또한 떨어지는 것이다. 제안 기법을 적용하여 공격을 방어하였을 때에는 normalized throughput이 떨어지기는 하지만 정도는 작다.

4.4 안전성 분석

4.4.1 송신자의 비밀키 추측 공격

송신자는 네트워크 코딩 패킷을 전달할 때 오염 공격을 방어하기 위해 HMAC을 사용하여 tag를 생성한다. 공격자가 오염 공격을 성공하기 위해서는 메시지 인증 절차를 통과해야 한다. 이를 위해 공격자는 수신한 tag에 사용되는 송신자의 비밀키를 추측하는 공격을 시도한다. 하지만 HMAC은 해쉬 함수와 동일한 성질을 갖고 있기 때문에 해쉬 함수의 일방향성으로 인해 송신자의 비밀키를 추측하는 것은 불가능하다.

송신자가 갖고 있는 비밀키를 k ($k \in 0,1^n$) 라고 하자. 이때 공격자가 임의로 생성한 키를 k'라고 할 때 $k = k'$ 이어야만 식 (8)이 성립한다.

$$Hash(k) = Hash(k') \tag{8}$$

하지만 tag를 생성하기 위해 HMAC SHA-256을 사용한다면, 해쉬값은 256비트가 된다. 따라서, 공격자는 비밀키를 추측하기 위해 256가지의 비밀키를 생성한다. 송신자의 비밀키와 동일한 키를 생성할 확률은 $\frac{1}{256}$이다. 즉 식 (9)처럼 송신자의 비밀키와 동일한 값을 생성하는 것은 거의 불가능하다.

$$Hash(k) \neq Hash(k') \tag{9}$$

4.4.2 메시지, tag 위조 공격

공격자는 송신자의 비밀키를 추측하는 시도 없이 임의의 메시지와 가짜 tag를 생성하여 메시지 인증 절차를 통과하는 공격을 시도한다. 공격자가 수신한 정보들로는 tag, 네트워크 코딩 패킷, 공개키가 있다. 공격자는 데이터 인증 절차를 통과하기 위해서 가짜 네트워크 코딩 패킷과 가짜 tag를 생성한다. 하지만 식 (10)처럼 가짜 tag와 네트워크 코딩 패킷, 공개키의 관계가 성립하지 않기 때문에 공격자의 메시지, tag 위조 공격은 성공할 수 없다.

$$\begin{aligned} h(z) \cdot M_f &\neq h(y) \cdot h(f) \cdot M_f \\ &\neq h(y) \cdot Tag_{M_f} \end{aligned} \tag{10}$$

4.4.3 Null 벡터 추측 공격

Null 벡터는 V가 S에게 네트워크 코딩 패킷 전달 권한을 요청할 때 인증 수단으로 사용된다. Null 벡터를 생성하기 위해서는 수신한 네트워크 코딩 패킷의 rank가 full rank가 되었을 때 가우시안 소거법으로 디코딩하여 원본 패킷을 얻어야 한다. 그 이후에 네트워크 코딩 패킷에 대한 null 벡터를 생성하여 S에게 전달한다. 이때 생성된 null 벡터는 1개 이상이 될 수 있다. 하지만 본 논문에서는 null 벡터의 전송 오버헤드를 최소화하기 위해 하나의 null 벡터만 전송하여 인증을 수행한다.

공격자는 원본 메시지를 갖고 있지 않지만 임의로 null 벡터를 생성하여 null 벡터 인증 절차를 통과하는 공격을 수행할 수 있다. 공격자가 null 벡터 인증 절차를 통과하기 위해 추측한 벡터 값이 실제 null 벡터와 일치할 확률은 다음과 같은 정리로 나타낼 수 있다.

정리 1. Finite field F^q 위의 N차원 벡터 공간에서 공격자가 임의의 null 벡터 x를 생성하여 인증 절차를 통과할 확률은 식 (11)과 같다.

$$\Pr(Px=0)=\sum_{i=1}^{N}(\prod_{k=1}^{i}\frac{1}{q^k})\frac{1}{i} \tag{11}$$

증명 : 정상적인 노드가 생성한 null 벡터 x의 차원은 N차원이다. 이때 field size는 q와 임의의 N을 추측하여 null 벡터와 일치할 확률은 $\frac{1}{q^N}$이 된다. 이때 공격자는 하나의 N에 대해서만 시도하는 것이 아니라 모든 N에 대해서 시도한다. N의 값을 1부터 N까지 계산하여 모든 확률의 합이 null 벡터 x가 원본 데이터와 내적을 하여 0이 되는 확률이 된다.

V. 결 론

본 논문에서는 무선 네트워크에서 하나의 소스 노드가 다수의 목적지 노드에게 멀티캐스트로 데이터를 전달할 때 중계 노드가 가상 소스의 역할을 함으로써 전체적인 네트워크 성능을 향상시킬 수 있는 방안을 제안하였다. 또한 네트워크 코딩이 적용된 환경이기 때문에 오염 공격을 방어하기 위한 데이터 인증과 노드 인증 기법을 제안하였다.

기존 논문에서 제안하고 있는 기법인 CodePipe에서는 목적지 노드가 가상 소스 역할을 하기 때문에 컨트롤 루프의 이점이 미비했다. 하지만 본 논문에서 제안한 기법은 코딩 포인트에서 가상 소스를 선정함으로써 CodePipe 보다 개선된 성능을 보이는 것을 시뮬레이션을 통해 확인할 수 있었다.

데이터 인증에 있어서는 기존 논문에서 제시한 기법인 KEPTE는 모든 N에 대한 키를 전송하기 때문에 전송 오버헤드, 저장 오버헤드가 발생하는 단점이 있었다. 하지만 제안 기법에서는 해쉬 체인 기법을 적용하여 첫 번째 키만 전송하고 다음 generation에 대한 키는 인증을 수행하는 노드들이 직접 생성함으로써 키 전송 오버헤드를 줄일 수 있었다.

마지막으로 가상 소스에 대한 인증 수단으로 null 벡터 인증 기법을 제안하였다. 가상 소스 선정 시 인증 기법을 적용한 사례가 없기 때문에 다른 논문과의 비교는 할 수 없었다. 하지만 시뮬레이션을 통해 제안 기법을 적용하였을 때 공격에 따른 방어효과를 볼 수 있었다.

References

[1] R. Ahlswede, N. Cai, S. Li, and R. Yeung, “Network information flow,” *Inf. Theory*, vol. 46, no. 4, pp. 1204-1216, Jul. 2000.

[2] T. Ho, M. Medard, R. Koetter, D. R. Karger, M. Effros, J. Shi, and B. Leong, “A random linear network coding approach to multicast,” *Inf. Theory,* vol. 52, no. 10, pp. 4413-4430, Oct. 2003.

[3] A. K. Haddad and R. H. Riedi, “Bounds on the benefit of network coding for wireless multicast and unicast,” *IEEE Trans. Mob. Comput.,* vol. 13, no. 1, pp. 102-115, Jan. 2014.

[4] K. H. Lee and J. H. Kim, “Random linear network coding to improve reliability in the satellite communication,” *J. KICS,* vol. 38B, no. 9, pp. 700-706, Sept. 2013.

[5] X. Yang, X. Tao, E. Dutkiewicz, E. X. Huang, Y. J. Guo, and Q. Cui, “Energy-efficient distributed data storage for wireless sensor networks based on compressed sensing and network coding,” *IEEE Trans. Wirel. Commun.,* vol. 12, no. 10, pp. 5087-5099, Oct. 2013.

[6] D. H. Lee, W. H. Lee, S. M. Kang, and H. Y. Hwang, “Frequency allocation and path selection scheme in underlay cognitive radio

networks using network coding," *J. KICS,* vol. 40, no. 12, pp. 2372-2380, Dec. 2015.

[7] P. Li, S. Guo, S. Yu, and A. V. Vasilakos, "Reliable multicast with pipelined network coding using opportunistic feeding and routing," *IEEE Trans. Parall. Distrib. Syst.,* vol. 25, no. 12, pp. 3264-3273, Dec. 2014.

[8] S. Chachulski, M. Jennings, S. Katti, and D. Katabi, "Trading structure for randomness in wireless opportunistic routing," in *Proc*. ACM SIGCOMM, pp. 169-180, 2007.

[9] R. Punnoose, P. Nikitin, and D. Stancil, "Efficient simulation of ricean fading within a packet simulator," in *Proc. IEEE 52nd Veh. Technol. Conf.*, pp. 764-767, 2000.

[10] E. Kehdi, et. al., "Null keys: Limiting malicious attacks via null space properties of network coding," in *Proc. Infocom,* pp. 1224-1232, Apr. 2009.

[11] A. Newell and C. Nita-Rotaru, "Split null keys: A null space based defense for pollution attacks in wireless network coding," in *Proc. SECON*, pp. 479-487, 2012.

[12] C. Chi, J. Tao, and Z. Qian, "TESLA-Based homomorphic MAC for authentication in P2P system for live streaming with network coding," *IEEE J. Sel. Areas in Commun.,* vol. 31, no. 9, pp. 291-298, Sept. 2013.

[13] Z. Rongfei, J. Yixin, L. Chuang, F. Yanfei, and S. S. Xuemin, "A distributed Fault/ Intrusion-Tolerant sensor data storage scheme based on network coding and homomorphic fingerprinting," *IEEE Trans. Parall. Distrib. Syst.,* vol. 23, no. 10, pp. 1819-1830, Oct. 2012.

[14] X. Wu, Y. Xu, C. Yuen, and L. Xiang, "A tag encoding scheme against pollution attack to linear network coding," *IEEE Trans. Parall. Distrib. Syst.,* vol. 25, no. 1, pp. 33-42, Jan. 2014.

[15] L. Lamport, "Password authentication with insecure communication," *Commun. ACM*, vol. 24, no. 11, pp. 770-772, Nov. 1981.

[16] J. Liu, et. al., "Efficient multicast key distribution using HOWP-based dynamic group access structures," *IEEE Trans. Computers*, vol. 62, no. 8, Aug. 2013.

안 명 기 (Myeong-Gi Ahn)

2014년 8월 : 아주대학교 정보컴퓨터공학과 학사
2017년 2월 : 아주대학교 컴퓨터공학과 석사
2017년 1월~현재 : LIG넥스원
<관심분야> 네트워크 코딩, 네트워크 보안, IoT

조 영 종 (Young-Jong Cho)

1985년 2월 : KAIST 석사
1990년 2월 : KAIST 박사
1996년 3월~현재 : 아주대학교 교수
<관심분야> 기계학습 활용, 무선 네트워크, 트래픽 모델링, 네트워크 코딩

강 경 란 (Kyungran Kang)

1994년 2월 : KAIST 석사
1999년 2월 : KAIST 박사
2004년 3월~현재 : 아주대학교 교수
<관심분야> 멀티캐스트, 이동 네트워크, 네트워크 코딩