한국정보전자통신기술학회논문지(jkiiect)'17-08, Vol.10 No.4
논문 17-10-04-340 http://dx.doi.org/10.17661/jkiiect.2017.10.4.340

해시 체인 기반 일회용 키를 이용한 하둡 보안 프로토콜 설계

정은희*, 이병관**

A Design of Hadoop Security Protocol using One Time Key based on Hash-chain

Eun-Hee Jeong*, Byung-Kwan Lee**

요 약 본 논문에서는 재전송공격과 가장 공격을 방지할 수 있는 하둡 보안 프로토콜을 제안한다. 제안하는 하둡 보안 프로토콜은 사용자 인증 모듈, 공개키 기반 데이터노드 관리 모듈, 네임노드 인증 모듈, 그리고 데이터노드 인증 모듈로 구성된다. 사용자 인증모듈은 인증서버에서 사용자의 신분을 확인한 후에 TGS로부터 임시접속 ID를 발급받고, 공개키 기반 데이터 노드 관리 모듈은 네임노드와 데이터 노드간의 비밀키를 생성하고, 해시체인기법으로 OKTL(One-Time Key List)를 생성한다. 네임노드 인증 모듈은 임시접속 ID로 사용자의 신분을 확인하고 DT(Delegation Token)와 BAT(Block Access Token)를 사용자에게 발급해주고, 데이터 노드 인증 모듈은 BAT의 OwnerID을 검증하여 사용자의 신분을 확인하고 데이터를 암호화시켜 사용자에게 제공한다. 즉, 제안하는 하둡 보안 프로토콜에서는 OTKL, timestamp, OwnerID을 이용하여 데이터 노드의 비밀키 노출을 대비할 뿐만 아니라 재전송 공격과 가장 공격을 탐지하고, 데이터 노드의 데이터 접근을 강화시키고 데이터를 암호화하여 전달함으로써 데이터 보안을 강화시켰다.

Abstract This paper is proposed Hadoop security protocol to protect a reply attack and impersonation attack. The proposed hadoop security protocol is consists of user authentication module, public key based data node authentication module, name node authentication module, and data node authentication module. The user authentication module is issued the temporary access ID from TGS after verifing user's identification on Authentication Server. The public key based data node authentication module generates secret key between name node and data node, and generates OTKL(One-Time Key List) using Hash-chain. The name node authentication module verifies user's identification using user's temporary access ID, and issues DT(Delegation Token) and BAT(Block Access Token) to user. The data node authentication module sends the encrypted data block to user after verifing user's identification using OwerID of BAT. Therefore the proposed hadoop security protocol dose not only prepare the exposure of data node's secret key by using OTKL, timestamp, owerID but also detect the reply attack and impersonation attack. Also, it enhances the data access of data node, and enforces data security by sending the encrypted data

Key Words : Authentication, Block Access Token, Delegation Token, Hadoop security, Hash chain, One Time Key

1. 서론

현재 다양한 클라우드 서비스를 이용하면서 클라우드 환경에 대한 인식이 높아졌을 뿐만 아니라 당연시 하는 경향이 생겨나기 시작하였다[1]. 그에 따라 클라우드 컴퓨팅 서비스를 제공하기 위해 기업 고유의 솔루션을 개발할 뿐만 아니라 오픈 스택[2]이나 유칼립터스[3], 하둡 프레임워크[4]와 같

This study was supported by 2015 Research Grant from Kangwon National University(No.201510076)
*Department of Regional Economics, Kangwon National University
**Corresponding Author : Department of Computer Engineering, Catholic Kwandong University (bklee@cku.ac.kr)
Received July 31, 2017 Revised August 04, 2017 Accepted August 09, 2017

은 오픈 소스 기반의 프레임워크도 활발히 연구 중이다[5,6].

특히, 하둡[7]은 아파치 너치(nutch)의 분산 처리를 지원하기 위해 개발된 것으로 구글의 빅데이터 관리 솔루션인 구글 파일 시스템(Google File System) 논문[8] 기반으로 구현된 GFS의 오픈 소스 버전이라고 할 수 있다. 하둡은 기존의 고가의 서버장비가 아닌 저렴한 일반 x86 CPU의 리눅스 장비에서 수 백 페타 바이트 이상의 데이터를 관리할 수 있도록 설계되었기 때문에 비용 부담이 적어 사용자가 빠르게 늘고 있으며, 페이스 북(Facebook), 아마존(Amazon), 이베이(ebay)를 포함한 수백 개의 기업에서 실제로 사용되고 있다[9,10].

초기에는 소수의 기업들만이 하둡을 사용하였으며 저장되는 데이터에 민감한 데이터의 비중이 매우 적었기 때문에 보안이 거의 도입되지 않았다. 하지만 사용자가 늘어남에 따라 하둡 분산 파일 시스템에 개인정보와 같이 보안을 필요로 하는 데이터의 양이 급격히 증가하기 시작하였고, 이에 하둡은 자신들의 기술에 보안을 도입할 필요성을 인식하여 2009년 사용자와 서버간의 강력한 상호 인증을 갖춘 새로운 버전[11]을 발표하였다[10,12].

이때, 하둡은 보안과 성능, 비용 모두를 만족하기 위하여 공개키 시스템은 배제하였고, 기존 하둡에 대칭키 시스템인 커버로스(Kerberos)를 결합하는 방법을 선택하였다. 하지만, 네임노드와 데이터 노드, 하둡 클라이언트가 모두 커버로스를 통해 인증하게 되면 커버로스의 키 분배 센터에 병목 현상이 발생하기 때문에 하둡은 성능 향상을 위해 자체적으로 개발한 대칭키 기반 토큰 시스템을 함께 도입하였다[13]. 하지만, 이렇게 보안이 도입된 하둡 분산 파일 시스템이라도 재전송 공격 및 가장 공격에 취약하며, 임의로 토큰을 생성하여 데이터에 접근하는 데이터 노드 해킹 공격이 가능하다는 문제점이 존재한다.

본 논문에서는 이러한 문제를 보완할 수 있는 하둡 보안 프로토콜을 제안한다. 제안하는 하둡 보안 프로토콜에서는 임시접근 ID로 사용자의 신분을 확인하고 해시체인기반 OTKL(One-Time Key List) 생성하고, 이 OTKL를 이용하여 데이터 블록 접근 토큰의 검증하도록 설계하였다. 그리고 일회성 암복호화키인 OTEK(One-Time Ecrypt Key)/OTDK (One-Time Decrypt Key)로 데이터를 암호화시켜 전달하도록 설계하여 재전송공격과 가장 공격을 방지할 뿐만 아니라 데이터 보안을 강화시키고자 한다.

2. 관련연구

2.1 커버로스

커버로스[10]는 티켓을 통하여 클라이언트와 서버간의 안전한 상호 인증을 제공하는 네트워크 인증 프로토콜이며, 신뢰할 수 있는 제 3자로서 인증에 사용되는 대칭키를 분배하는 키 분배 센터(Key Distribution Center)이기도 하다.

커버로스는 사용자를 인증하며, 사용자에게 사용자와 티켓 발급 서버 사이에 사용될 세션키를 발급하여 사용자와 티켓 발급 서버 간의 암호화 통신을 돕는 인증 서버(Authentication Server), 서비스 서버에 대한 사용자의 인증 값인 티켓을 사용자에게 발급해주는 티켓 발급 서버(Ticket Granting Server), 사용자에게 서비스를 제공하는 데이터 서버인 서비스 서버로 구성된다[12].

하둡에서는 클라이언트가 커버로스의 사용자가 되고, 네임노드와 데이터 노드가 커버로스의 서버가 된다. 하지만, 실제적으로는 네임노드가 커버로스의 서버 역할을 담당하며, 클라이언트와 네임 노드 사이를 인증한다. 커버로스를 이용한 하둡의 클라이언트와 네임 노드 사이의 인증철차는 그림 1과 같다.

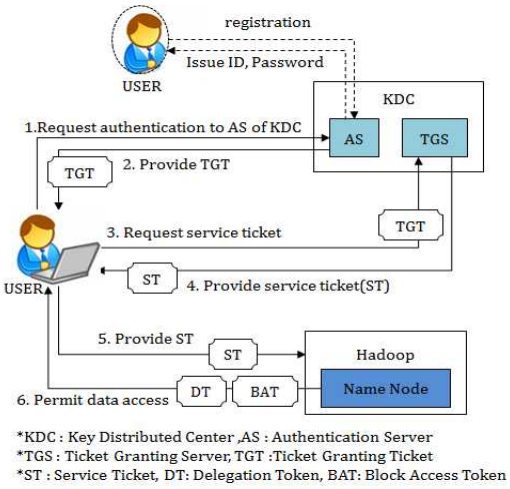

그림 1. 커버로스의 인증 과정
Fig. 1. The authentication process of Kerberos

2.2 토큰 시스템

2.2.1 위임 토큰

위임 토큰은 클라이언트와 네임노드가 인증을 위하여 공유하는 비밀 값이다. 클라이언트는 초기 커버로스 인증이 완료된 후 네임노드로부터 표 1과 같은 구조의 위임 토큰을 발행받고, 그 후의 네임노드에 대한 인증에 이를 사용한다. 네임노드는 발행한 위임 토큰들이 만료될 때까지 메모리에 보관한다.

표 1. 위임 토큰 구조[9]
Table 1. The structure of delegation token

Delegation Token	TokenID, Token Authenticator
TokenID	ownerID,renewerID,issueDate, maxDate,sequenceNumber
Token Authenticator	HMAC-SHA1(master Key, TokenID)

위임 토큰은 토큰 아이디(TokenID)와 토큰 인증자(Token Authenticator)로 구성된다. 토큰 아이디는 클라이언트 고유 식별자, 지정된 갱신자의 고유 식별자, 위임 토큰 발행일, 위임 토큰 만료일, 그리고 위임 토큰 생성 카운터로 구성된다. 토큰 인증자는 클라이언트를 인증할때 사용하는 값으로 master key와 토큰 아이디의 HMAC-SHA1값이다. 네임노드는 기본 10시간 주기로 마스터키를 랜덤하게 다시 생성한다.

2.2.2 블록 접근 토큰

블록 접근 토큰은 클라이언트가 네임노드로부터 데이터노드에 저장된 데이터에 대한 접근 권한을 얻기 위한 비밀값이다. 즉, 네임노드는 먼저 위임 토큰으로 클라이언트를 신분을 확인한 후에 클라이언트가 요청하는 블록에 대한 블록접근 토큰을 클라이언트에게 발행한다. 클라이언트는 이 블록접근 토큰을 이용하여 네임노드에 인증할 필요 없이 일정 시간 동안 비교적 간단하게 데이터노드에 접근할 수 있다.

네임노드는 위임 토큰에서 사용하는 마스터 키와 별개로 블록 접근 토큰 생성에 사용될 또 다른 랜덤 키를 주기적으로 생성하며, 랜덤 키는 기본 10시간을 주기로 생성된다. 네임노드는 하트비트 메시지와 함께 랜덤 키를 데이터노드들에게 전달한다. 블록 접근 토큰은 토큰 아이디와 토큰 인증자로 구성된다. 구조는 표 2와 같다[10].

토큰 아이디는 토큰의 만료일(expirationDate), 블록 접근 토큰 생성 시 사용되는 랜덤 키의 고유 식별자인 키 아이디(keyID), 클라이언트의 고유 식별자인 소유자 아이디(ownerID), 접근 요청한 블록의 위치를 나타내는 블록 아이디(blockID), 접근 모드(accessModes)로 이루어져 있다. 접근 모드는 접근 요청한 블록에 대한 읽기, 쓰기, 복사, 재배치 권한을 명시하는 값이다. 토큰 인증자는 네임노드와 데이터노드가 공유한 키와 토큰 아이디의 HMAC-SHA1 값을 의미한다[12].

표 2. 블록 접근 토큰 구조[9]
Table 2. The structure of block access token

Block Access Token	TokenID, Token Authenticator
TokenID	expirationDate, keyID, ownerID, blockID, accessModes
Token Authenticator	HMAC-SHA1(key, TokenID)

3. 하둡 보안 프로토콜 설계

본 논문에서 하둡의 네임 노드와 데이터 노드, 클라이언트 사이의 상호 인증 및 데이터 암호화하여 하둡의 데이터를 보호하는 하둡 보안 프로토콜을 제안하여 비밀키 노출로 인한 데이터 노드의 피해를 최소화시키고, 재전송 공격과 가장 공격을 차단하고자 한다. 그림 2는 제안하는 하둡 보안 프로토콜의 모듈 구성도와 전체 흐름도를 설명한 것이다.

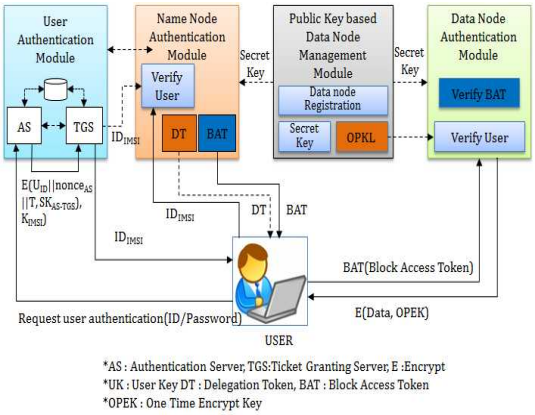

그림 2. 하둡 보안 프로토콜의 흐름도
Fig. 2. The flow chart of hadoop security protocol

제안하는 하둡 보안 프로토콜은 하둡 분산 파일 시스템에 데이터 저장하거나 읽고자 하는 사용자에게 사용자 인증키를 발급하는 사용자 인증 모듈, 네임노드와 데이터 노드 사이의 비밀키를 관리하는 공개키 기반 데이터 노드 관리 모듈, 하둡 분산 파일 시스템에서 데이터의 read/write를 요청하는 사용자의 신분을 확인하고 사용자가 하둡 분산 파일 시스템의 회원이면 데이터 블록 접근 토큰을 발급하는 네임 노드 인증 모듈, 그리고 사용자가 제시한 블록 접근 토큰을 확인한 후, 네임 노드가 발급한 블록 접근 토큰이면 사용자에게 데이터 블록을 제공하는 데이터 노드 인증 모듈로 구성된다.

본 논문에서 하둡 분산 시스템의 구성 요소인 인증서버(AS), 티켓 발행 서버(TGS), 네임 노드, 데이터노드는 각각 자신의 비밀키와 공개키를 가지고 있으며, 인증서버(AS)와 TGS 사이에는 공유 비밀키인 SK_{AS-TGS}가 등록되어 있다고 가정한다. 또한 사용자는 하둡 파일 시스템을 사용하기전에 사전에 ID와 패스워드가 이미 인증 서버에 등록해야 한다고 가정한다.

표 3. 표기법
Table 3. Notification

용어	의미
ID_U	User ID
ID_N	Name node ID
ID_{IMSI}	Temporary ID
AS_{ID}	Authentication Server ID
TGS_{ID}	Tikcet Generation Server ID
$SKey_N$	Name node's secret key
$PKey_N$	Name node's public key
$SKey_D$	Data node's screte key
$PKey_D$	Data node's public key
K_{IMSI}	Temporary Key
SK_{AS-TGS}	Sesseion key between authentication server and TGS
SK_{N-D}	Session key between name node and data node (=SK_{D-N})
H_{TGS}	Hash value of TGS
H_D	Hash value of data node
H_N	Hash value of name node
E_{AS}	Encrypted value using SK_{AS-TGS}
E_{TGS}	Encrypted value using $PKey_N$
$nonce$	Random number
OTKL	One Time Key List
OTEK	One Time Encrypt Key
OTDK	One Time Decrypt Key
•	multiply operation symbol
||	concatenation symbol

3.1 사용자 인증 모듈 설계

하둡 분산 파일 시스템에는 개인정보와 같이 보안을 필요로 하는 데이터가 존재하므로, 하둡 분

산 파일 시스템에 접속하여 데이터를 읽고 저장하려는 모든 사용자들의 신분 확인이 필요하다.

본 논문에서는 네임노드와 데이터 노드에서 각각 사용자 인증에 활용할 수 있는 사용자 인증키를 생성하는 사용자 인증 모듈을 설계하였다.

그림 3은 본 논문에서 제안하는 사용자 인증 모듈의 처리 절차를 설명한 것이며, 단계별 절차는 다음과 같다.

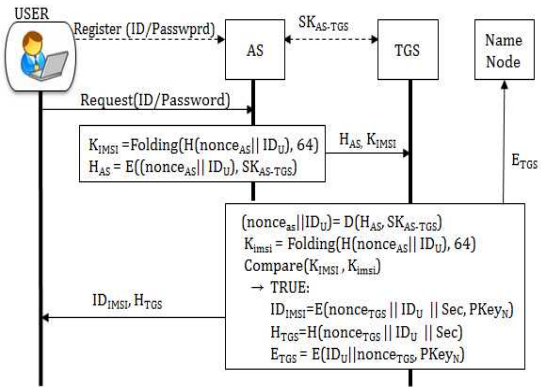

그림 3. 사용자 인증 모듈의 처리 과정
Fig. 3. The process of user authentication module

[1 단계] 하둡 분산 파일 시스템에 회원인 사용자는 ID와 Password로 인증서버에 사용자 인증을 요청한다.

[2 단계] 인증서버는 사용자의 ID와 Password를 확인한 결과, 인증서버에 등록된 사용자이면, 랜덤한 수 $nonce_{AS}$와 사용자 ID(ID_U)를 해싱한 후에 Folding 연산하여 64bit 임시 비밀키를 생성한다. 그리고 TGS와의 공유비밀키인 세션키(SK_{AS-TGS})로 사용자 ID와 $nonce_{AS}$를 암호화시킨 후에 임시 키인 K_{IMSI}와 암호문(E_{AS})을 TGS에게 전달한다.

$$K_{IMSI} = Folding(H(nonce_{AS} \| ID_U), 64)$$

$$E_{AS} = E((ID_U \| nonce_{AS}), SK_{AS-TGS})$$

[3 단계] TGS는 인증서버로부터 전달받은 암호문을세션키인 SK_{AS-TGS}로 복호화시킨다. 그리고 사용자 ID와 $nonce_{AS}$를 해시하고 Folding연산으로 64bit인 K_{imsi}를 계산한다.

$$(nonce_{TGS} \| ID_U) = D(H_{AS}, SK_{AS-TGS})$$

$$K_{imsi} = Folding(H(nonce_{AS} \| ID_U), 64)$$

[4 단계] TGS는 2단계에서 전달받은 K_{IMSI}와 3단계에서 산출한 K_{imsi}를 비교하여 전달받은 메시지의 무결성은 검증한다. 검증결과가 TRUE이면, TGS는 네임서버의 공유키인 $PKey_N$로 사용자의 ID를 암호화시켜 전달한다.

$$E_{TGS} = E(U_{ID} \| nonce_{TGS}, PKey_N)$$

[5 단계] TGS는 랜덤한 수 $nonce_{TGS}$, 사용자 ID, 임시 접속 ID의 유효시간(sec)을 네임노드의 공개키인 $Pkey_N$로 암호화시켜 임시 접속 ID를 생성한다. 그리고 TGS는 네임노드의 임시 접속 ID의 무결성 검증을 위한 해시값(H_{TGS})를 생성하여 사용자에게 전달한다.

$$ID_{IMSI} = E(nonce_{TGS} \| ID_U \| \text{sec}, PKey_N)$$

$$H_{TGS} = H(nonce_{TGS} \| ID_U \| \text{sec})$$

3.2 공개키 기반 데이터 노드 관리 모듈 설계

본 논문에서는 하둡 분산 파일 시스템의 서버 역할을 하는 네임 노드가 데이터가 분산 저장되는 데이터 노드를 관리하는 공개키 기반 데이터 노드 관리 모듈을 설계한다. 즉, 공개키 기반 데이터 노드 관리 모듈은 데이터 노드를 네임 노드에 등록하고, 네임 노드와 데이터 노드 사이의 비밀키를 생성한다. 이 비밀키는 차후에 블록접근토큰에 사용하여, 블록접근토큰을 검증하는데 사용한다.

그림 4는 공개키 기반 데이터 노드 관리 모듈을 절차를 설명한 것이고, 그 결과 값인 네임 노드와 데이터 노드는 동일한 OTKL(One Time Key List)를 생성한다.

[1 단계] 하둡 분산 파일 시스템의 새로운 데이터 저장소로 추가되는 데이터 노드는 데이터 노드의 ID를 네임 노드에 등록한다. 이때, 데이터 노드는 데이터 노드 ID(ID_D), 데이터 노드의 공개키($PKey_D$), 랜덤한 수인 $nonce_D$를 네임노드에 전달한다.

[2 단계] 네임노드는 수신한 데이터 노드 ID와 nonce를 연접하여 해시함수로 해시값(H_N)을 계산한 후, 네임노드 ID, 네임노드 공개키($PKey_N$), 해시값(H_N)를 데이터 노드에 전달한다.

$H_N = Hash(ID_D \| nonce_D)$

[3 단계] 데이터 노드는 자신의 ID와 $nonce_D$ 값을 연접하여 해시함수로 해시값(H_D)를 계산하고, 네임 노드로부터 전달받은 해시값(H_N)와 비교한다. 비교 결과가 참이면, 데이터 노드는 네임노드의 공개키($PKey_N$)에 자신의 비밀키($SKey_D$)를 곱셈 연산하여 세션키(Session Key)를 생성하고 세션키를 네임노드에 전달한다.

$H_D = Hash(ID_D \| nonce)$

$SK_{D-N} = PKey_N \cdot SKey_D$

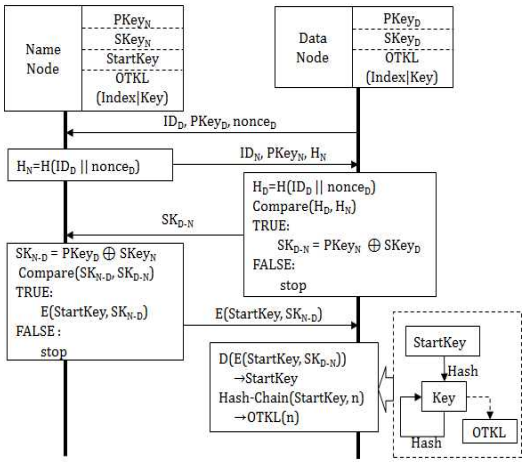

그림 4. 공개키 기반 데이터 노드 관리 모듈 처리 과정
Fig. 4. The process of data node management module based on public key

[4 단계] 네임노드는 1단계에서 수신한 데이터 노드의 공개키($PKey_D$)와 네임노드의 비밀키($SKey_N$)를 곱셈 연산하여 세션키(Session Key)를 생성한다. 그리고 3단계에서 전달받은 데이터 노드의 세션키와 비교한다. 그리고 비교 결과가 참이면, 네임노드는 Hash-chain기법으로 생성할 One Time Key의 초기값인 StartKey를 랜덤한 수로 생성하고, 이 StartKey를 세션키로 암호화하여 데이터 노드에 전달한다.

$SK_{N-D} = PKey_D \cdot SKey_N$

E(StartKey, SK_{N-D})

[5 단계] 데이터 노드는 세션키로 수신한 암호문을 복호화하여 StartKey를 획득하고, 이 StartKey를 해시체인기법으로 연산하여 OTKL를 계산하여 데이터 노드의 저장소에 보관한다.

D(E(StartKey, SN_{D-N}))

OTKL(Hash-chain(StartKey))

3.3 네임 노드 인증 모듈 설계

본 논문에서는 하둡 분산 파일 시스템에 접속하려는 사용자는 3.1절에서 취득한 임시 접속 ID로 네임노드에 접속하도록 설계한다. 네임노드는 이 임시 접속 ID로 사용자의 신분을 확인하고 위임토큰과 블록접근토큰을 생성하여 사용자에게 전달하도록 설계한다. 특히, 본 논문에서는 3.2절에서 생성된 OTKL (One Time Key List)에 저장된 Key의 색인 값을 블록접근토큰의 KeyID로 사용하도록 설계하여 데이터 노드가 블록접근토큰을 검증하는데 사용함으로써, 재전송 공격, 가장 공격을 탐지하도록 설계한다.

그림 5는 네임 노드와 사용자간의 상호 인증 절차를 설명한 것이며, 네임노드의 상호 인증 절차와 네임노드의 위임 토큰(Delegation Token), 블록접근토큰(Block Access Token) 생성의 단계별 과정은 다음과 같다.

[1 단계] 사용자는 TGS로부터 발급받은 임시 ID (ID_{IMSI}), 임시 ID에 대한 해시값(H_{TGS})를 해당 네임노드에 전달한다.

[2 단계] 네임노드는 TGS로부터 사용자의 ID (ID_U)를 암호화시킨 E_{TGS}를 전달받았고, 네임노드는 자신의 비밀키로 E_{TGS}를 복호화시키고, 사용자로부터 전달받은 임시 ID(ID_{IMSI})를 네임노드의 비밀키로 복호화한다.

$ID_u \| nonce_{TGS} = D(E_{TGS}, SKey_N)$

$nonce_{TGA} \| ID_U \| Sec = D(ID_{IMSI}, SKey_N)$

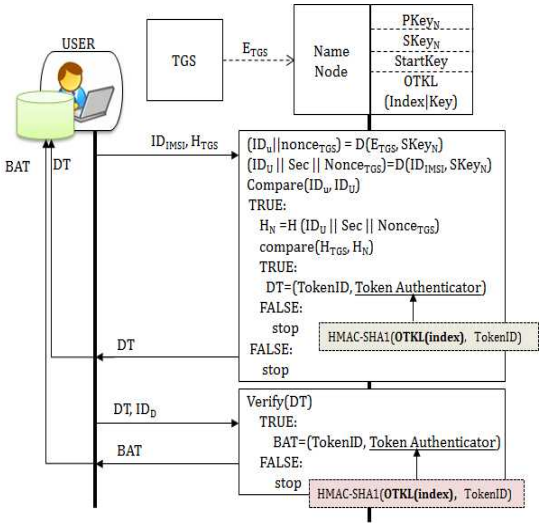

그림 5.네임 노드 인증 모듈 처리 과정
Fig. 5. The process of name node authentication module

[3 단계] 네임노드는 2단계의 결과값에서 각각 ID_u와 ID_U를 추출하여 비교한 결과가 참이고 유효시간이내이면, 네임노드는 복호화된 $nonce_{TGA}||ID_U||Sec$를 해시함수로 해시값(H_N)을 계산한다. 그리고 사용자가 전달한 해시값인 H_{TGS}와 H_N를 비교하여 참이면, 네임노드는 사용자가 전달한 위임토큰(Delegation Token)을 생성하여 사용자에게 전달한다.

$$H_N = Hash(nonce_{TGA}||ID_U||Sec)$$

DT=(TokenID, Token Authenticator)

TokenID=(ownerID,renewerID,issueDate,maxDate,sequenceNumber)
ownerID = ID_U
renewerID = ID_N
issueDate = Timestamp
maxDate = Timestamp + 10(hour)
seqeunceNumber=sequenceNumber+1
Token Authenticator
=HMAC-SHA1(OTKL(index), TokenID)

[4 단계] 사용자는 위임토큰을 이용하여 데이터 노드와 네임 노드에 데이터 접근을 요청한다. 이때, 네임 노드는 사용자가 전달한 위임토큰을 토큰인증자로 검증하고, 검증결과가 참이면 사용자 ID가 요청한 데이터 블록에 대한 데이터 접근 권한, 블록접근토큰 유효기간, 소유자 ID등을 결정한 후, 블록접근토큰(Block Access Token)을 생성하여 사용자에게 전달한다.

BAT=(TokenID, Token Authenticator)

TokenID = (expirationDate, KeyID, ownerID, blockID, accessModes)
expirationDate = Timestamp + 10(hour)
KeyID = OTKL(index)
OwnerID = ID_U
blockID = {1, 2, ... , n}
accessModes = [read only|read|write|..]
Token Authenticator
= HMAC-SHA1(OTKL(index), TokenID)

3.4 데이터 노드 인증 모듈 설계

본 논문에서는 네임 노드로부터 위임 토큰을 발급받은 인증된 사용자는 데이터 노드에 있는 데이터 블록을 가져오거나 저장하기 위해 BAT를 데이터 노드에 전달한다. 이때 모든 데이터 노드들은 3.2절에서 설계한 OTKL를 가지고 있으며, BAT를 수신한 데이터 노드는 자신이 보유하고 있는 OTKL의 Key를 이용하여 BAT의 무결성을 검증하고 데이터 재전송 공격을 탐지하는 데이터 노드 인증 모듈을 설계한다.

그림 6은 데이터 노드 인증 모듈의 처리 과정을 설명한 것이며, 단계별 절차는 다음과 같다.

[1 단계] 사용자는 사용자 ID, 네임노드로부터 발급받은 블록접근토큰(Block Access Token), 데이터 블록 번호, 랜덤한 수 $nonce_U$, $tampstamp_U$를 데이터 노드에 전달하고, 데이터를 복호화하기 위한 OTDK(One Time Decrypt Key)를 생성한다.

$$OTDK = Folding(H(OwnerID || nonce_U || Timestamp_U), 64bit)$$

[2 단계] 데이터 노드는 사용자로부터 수신한 블록접근토큰이 유효기간이내의 블록접근토큰인지, 블록 번호가 존재하는지, 그리고 OwnerID가 맞는

지를 확인한다. 이때, 이 모든 확인 결과가 참이어야만 다음 단계를 진행하고, 그렇지 않으면 사용자에게 블록접근토큰이 유효하지 않다는 메시지를 전달한다.

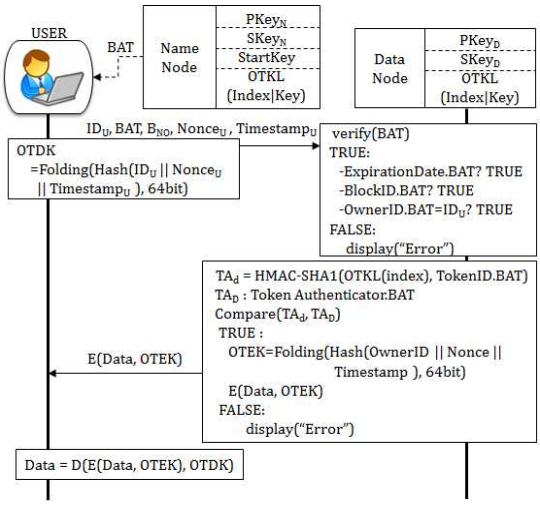

그림 6. 데이터 노드 인증 모듈 처리 과정
Fig. 6. The process of data node authentication module

[3 단계] 데이터 노드는 BAT의 KeyID= OTKL(index)에 해당하는 Key값을 가져와서 HMAC-SHA1로 해시값을 계산하고, BAT의 Token Authenticator와 비교하여 BAT의 무결성을 검증한다. 이때, 무결성 검증 결과가 참이면, BAT와 사용자가 전달한 정보를 이용하여 데이터를 암호화하기 위한 OTEK(One Time Encrypt Key)를 생성하고 데이터를 암호화하여 사용자에게 전달한다. 무결성 검증 결과가 거짓이면, 사용자에게 블록접근토큰이 유효하지 않다는 메시지를 전달한다.

TA_D = HMAC-SHA1(OTKL(index), TokenID)

OTEK = Folding(H(OwnerID || $nonce_U$ || $Timestamp_U$), 64bit)

E(DataBlock, OTEK)

[4 단계] 사용자는 데이터 노드로부터 전달받은 데이터를 OTDK로 복호화한다.

data = D(E(Data, OTEK), OTDK)

4. 분석

4.1 재전송 공격 탐지

기존의 위임 토큰과 블록접근토큰은 소유자 아이디에 대한 인증을 하지 않기 때문에 토큰 유효시간 동안에 재전송 공격을 시도할 수 있다.

제안하는 하둡 보안 프로토콜의 네임 노드 인증 모듈에서는 OTKL(One Time Key List)에 저장된 Key의 색인 값을 블록접근토큰의 KeyID로 사용하도록 설계하였다. 그리고 데이터 노드 인증 모듈에서는 데이터 노드가 블록접근토큰을 검증하는데 OTKL의 Key를 이용하여 BAT의 무결성을 검증함으로써 재전송 공격을 탐지할 수 있도록 하였다.

또한, 데이터 노드 인증 모듈에서는 BAT의 OwnerID을 검증함으로써 가장 공격을 탐지할 수 있으며, 네임노드는 OTKL에 있는 하나의 키 값으로 암호화하여 새로운 StartKey를 데이터 노드들에게 배포하여 주기적으로 OTKL를 갱신함으로써 OTKL의 노출로 인한 데이터 손실을 방지할 수 있다.

4.2 데이터 기밀성 유지

제안하는 하둡 보안 프로토콜에서는 1회용으로 사용자가 데이터 노드에 데이터 블록을 요청할 경우에만 일시적으로 작성하는 키로써 OTEK, OTDK를 생성한다. 이 OTEK, OTDK는 키 생성시의 timestamp를 이용하므로, 일정한 시간 이후에는 이 키를 이용하여 데이터를 암호화하거나 복호화할 수 없으므로 데이터의 기밀성을 유지할 수 있다.

특히, OTEK와 OTDK는 키를 서로 교환하는 것이 아니라 사용자와 데이터 노드가 각자 생성하고 Folding기법을 키를 생성하기 때문에 키의 안전성을 강화시켰을 뿐만 아니라 키 노출을 방지할 수 있다.

4.3 성능 분석

성능분석에서는 사용자가 데이터 블록에 접근을

요청하여 데이터를 전달받는 과정을 기존의 하둡 분산 파일 시스템과 제안하는 시스템을 비교하였다.

비교 결과, 총 라운드 수는 6회로 동일하지만 제안하는 기법에서는 네임노드는 공개키, 비밀키, OTKL, 그리고 StartKey를 사용하고, 데이터 노드는 공개키, 비밀키, OTKL, 그리고 OTEK을 사용하고, 사용자는 OTDK를 사용하도록 설계하였다. 즉, 기본의 하둡분산 파일 시스템의 네임노드의 마스터 키와 랜덤 키, 데이터 노드의 랜덤 키보다 키 개수가 많지만, 하둡 분산 파일 시스템에 기밀성을 유지하면서 재전송 공격을 방지할 수 있도록 하였다.

표 4. 성능분석
Table 4. Performance

		HDFS	Proposed scheme
Total round		6	6
The number of key	Name node	2	4
	Data node	1	4
	User	0	1
Stability	Reply attack	×	○
	Data encryption (Confidentiality)	×	○

5. 결론

본 논문에서 하둡의 네임 노드와 데이터 노드, 사용자 사이의 상호 인증 및 데이터 암호화하여 하둡의 데이터를 보호하는 하둡 보안 프로토콜을 제안하였다.

제안하는 하둡 보안 프로토콜은

첫째, 기본의 하둡 보안 기법인 커버로스, 위임토큰, 블록접근토큰을 활용하지만, 위임토큰과 블록접근토큰은 네임노드가 생성하여 사용자에게 전달하도록 설계하였다.

둘째, 해시 체인 기법을 이용하여 네임노드와 모든 데이터 노드가 동일한 OTKL를 갖도록 설계하였고, 주기적으로 OTKL를 갱신함으로써 OTKL의 노출로 인한 데이터 노출을 방지할 수 있다.

셋째, OTKL의 Key로 데이터 재전송 공격을 탐지하고, BAT의 OwnerID을 검증함으로써 가장 공격을 탐지할 수 있다.

넷째, 데이터 노드의 데이터 블록을 암호화하고 복호화하는 일회용 암복호 키를 사용자와 데이터 노드가 각각 생성하도록 설계하였고, 데이터 노드가 데이터를 암호화하여 사용자에게 전달함으로써 데이터의 노출을 방지할 수 있다.

REFERENCES

[1] Sung-Jae Jung, Yu-Mi Bae, “Trend analysis of Threats and Technololgies for Cloud Security,” Journal of Security Engineering, vol.10, no.2, pp.199-212, 2013. 4.

[2] Openstack, http://www.openstack.org/

[3] D. Nurmi, et al., “Eucalyptus: A Technical Report on an Elastic Utility computing Architecture Linking Your Programs to Useful System,” Technical Report 2008-10, UCSB Computer Science, 2008.

[4] K. Schvachko, et al., “The Hadoop Distributed File System,” in 26th IEEE Symposium on Massive Storage Systems and Technologies, May 2010.

[5] Wikipedia, http://en.wikipedia.org/wiki/Hadoop

[6] Seung-Je Park, Heeyoul Kim, “Improving Hadoop Security Through Hash-chain,” The Journal of Korean Institute of Information Technology, vol.10, no.6, pp.65-73, 2012.06.

[7] Apache Hadoop, http://hadoop.apache.org/

[8] S. Ghemawat, H. Gobioff, and S. Leung, “The google file system,” Proceedings of ACM Symposium on Operating Systems Principles, pp.29-43, Oct. 2003.

[9] T. White, “Hadoop: the definition guide,” O’Reilly edia, Yahoo! Press, Jun 2009.

[10] So Hyeon Park, Ik Rae Jeong, “A Study on Security Improvement in Hadoop Distributed File System Based on Kerberos,” Journal of The Korea Institute of Information Security & Cryptology(JKIISC), vol.23, no.5, pp.803-813, 2013.10.

[11] O. O'Malley, K. Zhang, S. Radia, R. Marti, C. Harrell, “Hadoop security design,” Oct. 2009. http://techcat.org/wp-content/uploads/2013/04/hadoop-security-design.pdf

[12] So Won Jeong, Kee Sung Kim, Ik Rae Jeong, “Secure Authentication Protocol in Hadoop Distributed File System based on Hash Chan,” Journal of The Korea Institute of Information Security & Cryptology(JKIISC), vol 23, no.5, pp.831-847, 2013.10.

[13] B. C. Beuman, T. Tso, “Kerberos: An authentication service for computer network,” IEEE Communications, vol. 32, no.9, pp. 33-38, Sep. 1994.

저자약력

정 은 희(Eun-Hee Jeong) [종신회원]

- 1998년 2월 : 관동대학교 일반대학원 전자계산공학과 (공학석사)
- 2003년 2월 : 관동대학교 일반대학원 전자계산공학과 (공학박사)
- 2003년 9월 ~ 현재 : 강원대학교 지역경제학과 교수

<관심분야>
전자상거래 보안, 빅 데이터, 헬스케어, IoT 보안

이 병 관(Byung-Kwan Lee) [종신회원]

- 1986년 2월 : 중앙대학교 전자계산공학과 (공학석사)
- 1990년 2월 : 중앙대학교 전자계산공학과 (공학박사)
- 1988년 3월 ~ 현재 : 가톨릭관동대학교 컴퓨터공학과 교수

<관심분야>
네트워크 보안, 빅 데이터, 데이터 마이닝, IoT 보안