KIPS Tr. Comp. and Comm. Sys.
Vol.4, No.4 pp.141~146 pISSN: 2287-5891

http://dx.doi.org/10.3745/KTCCS.2015.4.4.141

A Design of Permission Management System Based on Group Key in Hadoop Distributed File System

Hyungjoo Kim† · Jungho Kang†† · Hanna You††† · Moonseog Jun††††

ABSTRACT

Data have been increased enormously due to the development of IT technology such as recent smart equipments, social network services and streaming services. To meet these environments the technologies that can treat mass data have received attention, and the typical one is Hadoop. Hadoop is on the basis of open source, and it has been designed to be used at general purpose computers on the basis of Linux. To initial Hadoop nearly no security was introduced, but as the number of users increased data that need security increased and there appeared new version that introduced Kerberos and Token system in 2009. But in this method there was a problem that only one secret key can be used and access permission to blocks cannot be authenticated to each user, and there were weak points that replay attack and spoofing attack were possible. Hence, to supplement these weak points and to maintain efficiency a protocol on the basis of group key, in which users are authenticated in logical group and then this is reflected to token, is proposed in this paper. The result shows that it has solved the weak points and there is no big overhead in terms of efficiency.

Keywords : Authentication, Hadoop, HDFS, Kerberos, Group Key

하둡 분산 파일 시스템에서 그룹키 기반 Permission Management 시스템 설계

김 형 주† · 강 정 호†† · 유 한 나††† · 전 문 석††††

요 약

최근 스마트 기기 및 소셜 네트워크 서비스, 스트리밍 서비스 등 IT 기술의 발달로 인해 데이터가 급증하였다. 이러한 환경에 맞춰 대용량 데이터를 처리할 수 있는 기술도 함께 주목받고 있는데, 가장 대표적인 기술이 하둡이다. 하둡은 오픈 소스 기반으로 리눅스 기반의 범용 컴퓨터에서 실행할 수 있도록 설계되었다. 초기 하둡은 보안이 거의 도입되지 않았으나, 사용자가 늘어남에 따라 보안이 필요한 데이터가 증가하면서 2009년 커버로스와 토큰 시스템을 도입한 새로운 버전이 나왔다. 그러나 이 방식은 하나의 비밀키만을 사용하고, 사용자마다 블록에 대한 접근허가를 지정할 수 없다는 문제점과 재전송 공격 및 위장 공격 등이 가능하다는 취약점을 가지고 있다. 따라서 본 논문에서는 이러한 취약점을 보안하면서 성능을 유지하기 위해 사용자들을 논리적인 그룹으로 묶어서 인증하고, 이를 토큰에 반영하는 그룹키 기반의 프로토콜을 제안하였다. 성능평가 결과 키 생성에 따른 오버헤드가 없고, 비밀키 유출에 대한 취약점을 해결하였음을 확인하였다.

키워드 : 인증, 하둡, HDFS, 커버로스, 그룹키

1. 서 론

최근 소셜 네트워크 서비스와 대용량 멀티미디어 데이터의 이용자가 늘어남으로써 인터넷에서 사용되는 디지털 데이터의 양이 급격히 증가하고 있다. 하둡은 이와 같은 대규모의 데이터를 처리하기 위한 소프트웨어 프레임워크를 제공하는 오픈소스 프로젝트로서, 야후(Yahoo)와 페이스북(Facebook), 아마존(Amazon) 같은 기업에서 실제로 사용되고 있다[1].

하둡 사용자가 늘어남에 따라 하둡 분산 파일 시스템(HDFS, Hadoop Distributed File System)에 개인정보와 같이 보안을 필요로 하는 데이터 양이 급격히 증가하기 시작하였다. 이에 하둡은 보안 도입의 필요성을 인식하여 2009년

※ This work was supported by the ICT R&D program of MSIP/IITP. [R0112-14-1061, The analysis technology of a vulnerability on an open-source software, and the development of Platform]
† 준 회 원 : 숭실대학교 컴퓨터학과 박사과정
†† 준 회 원 : 숭실대학교 컴퓨터학과 박사
††† 준 회 원 : KT 융합연구소 연구원
†††† 종신회원 : 숭실대학교 컴퓨터학과 교수
Manuscript Received : February 10, 2015
First Revision : February 23, 2015
Accepted : February 23, 2015
* Corresponding Author : Moonseog Jun(mjun@ssu.ac.kr)

SASL(Simple Authentication and Security Layer)을[2] 사용한 새로운 버전을 발표하였다[3]. 하지만 이렇게 보안이 도입된 하둡은 키 문제, 재전송 공격, 위장공격 등에 취약점이 있다.

본 논문에서는 하둡 분산 파일 시스템이 가지고 있는 취약점을 분석하고, 3절에서 하둡 보안의 취약점을 개선하기 위해 그룹키를 이용한 새로운 인증 프로토콜을 제안한 후, 4절에서 그에 따른 안전성과 성능을 분석하였다.

2. 관련 연구

2.1 하둡

하둡(Hadoop)은 빅데이터를 처리하는 데 가장 널리 사용되는 솔루션으로, 하둡 분산 파일 시스템과 맵리듀스(MapReduce)를 구현한 것이다[4]. 하둡은 마스터/슬레이브 구조로 이루어져 있으며, Fig. 1과 같이 NameNode와 DataNode들로 구성되어있다. DataNode는 파일들을 64KB의 블록 단위로 분할하여 저장하고 있으며, 하드웨어 등의 문제로 데이터에 문제가 생길 것을 우려하여 하나의 블록은 다른 DataNode에 복사되어 저장된다. NameNode는 DataNode들에 저장된 파일의 정보와 Replicas 개수, 블록ID, 각 블록이 저장되어있는 DataNode의 위치 정보 같은 Metadata를 관리하며, 사용자로부터 블록의 읽기, 쓰기 등의 요청을 받아 서비스 하는 등 시스템상의 전반적인 일을 수행한다.

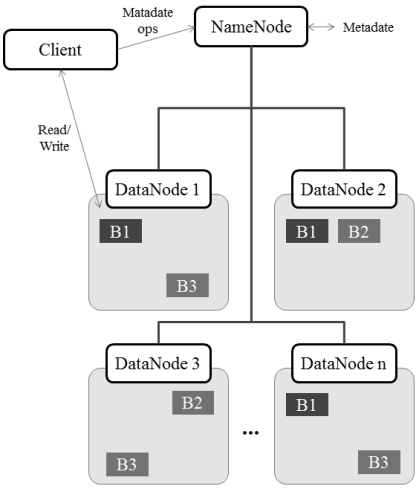


Fig. 1. Hadoop Architecture

2.2 하둡 보안 시스템

초기 하둡은 클라이언트 인증과 블록에 대한 권한 제어 등과 같은 보안 기능을 제공하지 않았다. 따라서 공격자가 블록 ID만 알면 클라이언트를 가장하여 블록에 접근할 수 있다는 취약점이 있었다[3].

하지만 하둡을 사용하는 사용자가 늘어나고 하둡에 저장되는 민감 정보도 증가하기 시작하면서 보안을 도입하기 시작하였고, 2009년 클라이언트와 NameNode 사이에 상호인증을 갖춘 공개키 기반의 보안 서비스인 SSL(Secure Socket Layer)을 적용하였다[3]. 하지만 공개키 기반의 계산을 통한 암·복호화 오버헤드로 인해, 대용량의 데이터를 빠르게 처리해야 하는 빅데이터 환경에 적합하지 않아 현재 거의 사용하지 않는다[5]. 그 후, 하둡 버전 1.0에서 GSS-API (Generic Services Application Program Interface)를 통하여 RPC(Remote Procedure Call)와 RPC 레벨에서 구현되고 있는 자바의 SASL(Simple Authentication and Security Layer)을 채택하였으며[3], 현재 하둡 SASL은 대칭키 기반의 커버로스와 DIGEST-MD5를 통한 클라이언트 인증을 지원한다[5].

2.3 하둡 보안 취약점

1) 하나의 비밀키 사용에 따른 데이터 유출 취약점

하둡 분산 파일 시스템에서는 DataNode에 저장된 파일을 보호하기 위해 커버로스와 토큰 시스템을 적용하고 있다. 토큰 시스템은 클라이언트가 DataNode에 저장된 파일을 읽기, 쓰기, 수정 등을 하기 위해 NameNode로부터 토큰을 받아서 DataNode에 사용하는 것으로 이를 블록 접근 토큰(BAT, Block Access Token)이라 한다. 블록 접근 토큰을 사용함으로써 DataNode는 클라이언트에 대한 접근 제어 및 인증을 제공할 수 있다.

사용자가 파일을 읽기 위해서 NameNode에 파일을 구성하고 있는 블록 정보와 DataNode의 위치를 요청한다. NameNode는 Metadata를 이용하여 사용자가 블록에 접근할 수 있도록 블록 접근 토큰을 전송한다.

블록 접근 토큰은 Table 1과 같이 토큰ID와 토큰 Authenticator로 구성되어있다. 토큰 ID는 토큰의 만료일(expirationDate), 토큰 Authenticator를 만들기 위해 사용되는 키값을 식별할 수 있는 키 아이디(keyID), 토큰의 소유자를 나타내는 소유자 아이디(ownerID), 토큰으로 접근할 수 있는 블록 아이디(blockID), 블록에 대한 읽기, 쓰기, 복사, 대체 중 어떤 것을 수행할 것인지를 나타내는 접근 모드(accessModes)로 이루어져 있으며, 토큰 Authenticator는 토큰ID와 NameNode와 DataNode 사이에 공유된 비밀키값을 이용하여 해시 연산한 값으로 이루어져 있다[5].

이때, 사용하는 비밀키값은 NameNode와 DataNode들 사이에 하나의 비밀키값을 공유하기 때문에, 비밀키가 노출된다면 모든 DataNode의 데이터들이 유출될 수 있다[6].

Table 1. Format of Block Access Token

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

2) 토큰 재전송 공격 및 위장공격

현재 블록 접근 토큰은 토큰 ID와 이용하여 비교하는 것 이외에 실제 토큰 사용자에 대한 검증 과정이 없다[3]. 이러

한 문제점은 사용자가 발급받은 블록 접근 토큰을 공격자가 중간에서 가로챈 후 토큰의 만료기간 내에 재전송 공격 또는 위장공격을 가능하게 한다.

NameNode가 사용자에게 블록 접근 토큰을 전송할 때 공격자가 중간에 가로챌 수 있으며, 이때 가로챈 블록 접근 토큰을 이용하여 사용자인 척 위장하여 DataNode에 데이터를 요청할 수 있다. 또한 사용자가 DataNode에 데이터를 요청할 때 전송하는 블록 접근 토큰을 가로챈 후, 이 토큰을 재전송하여 DataNode에 데이터를 요청할 수 있다. 이렇게 공격자가 전송한 블록 접근 토큰은 변경된 것이 아니기 때문에 DataNode는 아무런 제지 없이 데이터를 전송하게 된다[7].

3) 블록에 대한 Permissions 미구축

사용자는 NameNode에 요청을 하여 DataNode에 있는 데이터를 읽거나, 블록에 데이터를 쓰기 등을 할 수 있다. 그러나 하둡에 인가된 특정 사용자가 DataNode에 저장된 데이터에 접근 권한이 있는지, 또는 데이터를 읽을 수만 있고, 수정 또는 대체를 할 수는 있는지 등을 구분하는 접근 허가(Permissions)을 제공하고 있지 않다[8]. 이러한 점은 공격자가 인가된 사용자의 로그인정보나 세션을 탈취해 DataNode의 모든 데이터를 읽거나 악의적으로 수정, 삭제할 수 있다는 문제점이 있다.

3. 제안하는 시스템

본 논문의 2.3에서 설명한 하둡 분산 파일 시스템에서 블록 접근 토큰의 재사용 공격과 블록에 대한 접근 권한을 확인하지 않는 취약점을 해결하고자, 사용자 그룹키를 이용한 하둡 분산 파일 시스템을 제안한다. 데이터 블록에 접근 권한을 부여하기 위해서는 다음과 같은 요구사항이 필요하다.

① 기존의 하둡 분산 파일 시스템과의 호환성을 높이기 위해 기존 구조를 유지한다.
② 파일 읽기와 쓰기에 따라 다른 권한을 부여할 수 있어야 한다.
③ 파일을 생성한 사용자와 같은 그룹에 속한 타 사용자는 데이터 블록에 접근할 수 있다.
④ 사용자의 하위 그룹에 속한 타 사용자는 사용자가 별도로 접근 권한을 변경하지 않는 이상 사용자가 생성한 데이터 블록에 접근할 수 없다. 예를 들어, 과장이 생성한 문서 파일에 일반 사원은 접근할 수 없다.

3.1 제안하는 시스템 구조

본 논문에서는 사용자가 파일을 읽고, 쓰기 위해 NameNode에 요청을 할 때, 사용자가 자신의 키값과 자신이 속한 그룹 키값을 이용하여 권한을 확인받을 수 있도록 한다. 사용자 인증은 기존의 NameNode에서 이루어져도 되지만, 본 논문에서는 키 관리 및 처리의 효율을 위해 Fig. 2와 같이 KeyNode를 추가하였다. KeyNode는 사용자들의 ID와 사용자 키값을 관리하며 그룹 ID와 그룹키값을 생성하고, 관리하는 역할을 수행한다.

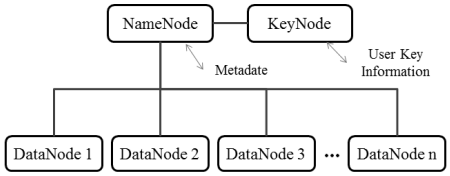


Fig. 2. Proposed System Architecture

KeyNode에서 관리하는 사용자 키값과 그룹키값들은 Fig. 3과 같이 논리적인 트리형태를 유지한다.

사용자 $U = U_1, U_2, U_3, \ldots U_n$는 각 자신의 키값 UK_n를 가지고 있으며, 하나의 그룹 G에 속해 있다. 그룹키 GK_n은 그룹에 속한 사용자들의 키값을 이용하여 생성되며, 생성된 그룹키는 해당 그룹의 사용자에게 전송되어 사용자가 그룹키를 가지고 있을 수 있도록 한다.

새로운 사용자 U_m이 가입하면 사용자는 자신의 키값 UK_m을 생성하고, 사용자가 속할 그룹 G_{AA}를 결정하여 KeyNode에게 알린다. KeyNode는 사용자 정보와 사용자 키값 UK_m을 등록하고, 해당 그룹의 키 GK_{AA}와 그의 조상 그룹키 GK_A를 변경한다.

$$GK_{AA} = f(g(UK_1), g(UK_2), g(UK_m)) \quad (1)$$
$$GK_A = f(g(GK_{AA}), g(UK_3), g(UK_4))$$

f와 g는 충돌회피 해시함수(CRHF, Collision-Resistant Hash Function)이다. 이와 같이 변경된 그룹키는 각 사용자와 공유하고 있는 비밀키로 암호화하여 사용자에게 전송된다. 사용자가 자신의 키값을 변경하거나, 탈퇴할 때에도 동일한 방법으로 해당 그룹의 키값을 변경하고 공유한다.

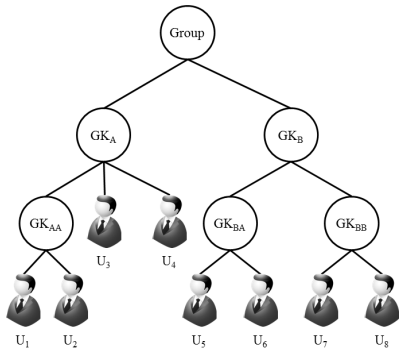


Fig. 3. User Group Architecture

3.2 그룹키를 이용한 데이터 접근 방식

사용자는 파일을 NameNode에 요청을 하여 블록 단위로 나눈 후에, DataNode에 해당 블록을 저장하고, 읽을 수 있다. NameNode는 한 파일의 정보와 파일이 어떤 블록으로 나뉘어져 있으며, 블록들이 어느 DataNode에 저장되어있는지에 대한

Metadata를 가지고 있다. 본 논문에서는 파일의 접근 권한을 확인하기 위해 파일의 Metadata에 사용자의 형제 그룹(EqualGroup)과 조상 그룹(HighGroup), 기타 사용자(OtherGroup)에게 허용 가능한 접근 모드를 명시하도록 한다.

$$Metadate = filename, file-info, fileownerID \quad (2)$$
$$EqualGroup, HighGroup, OtherGroup,$$
$$numReplicas, Block-ids,$$
$$Block-id-Locations$$

사용자는 파일을 읽기 위해서 Fig. 4와 같이 동작한다.

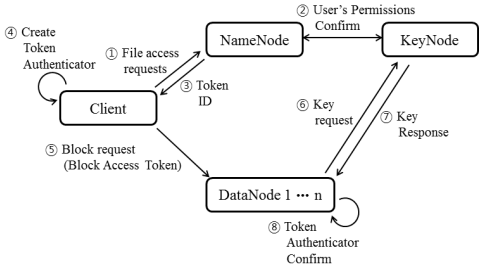


Fig. 4. Read Operation Sequence of Block

우선, 사용자는 NameNode에 파일 접근 요청메시지(FAM)를 전송한다. 이때, 사용자는 자신의 ID와 인증받을 수 있는 값을 함께 전송한다.

$$FAM = FA_{id}(fileinfo, userID, groupID, accessModes), \quad (3)$$
$$H(U_{key}, FA_{id})$$

NameNode는 KeyNode에게 FAM을 전송하여 사용자를 인증받은 후, Metadata를 이용하여 사용자가 파일을 AccessModes에 따라 접근할 수 있는 권한이 있는지 확인한다.

Table 2. Proposed Format of Block Access Token

Title	Format
TokenID	ExpirationDate, keyID, ownerID, blockID, AccessModes, AccessPermission_ID
Token Authenticator	HMAC-SHA1(AccessKey, HMAC-SHA1(Key, TokenID))
Block Access Token	TokenID, TokenAuthenticator

접근 권한이 확인되면 TokenID에는 Access Permission_ID 값을 추가하여 블록 접근 토큰을 발행한다. Access Permission_ID는 해당 사용자의 접근 권한이 파일 소유자인지 형제 그룹이거나 조상 그룹인지, 그룹 사용자인지 여부를 확인한다. 블록 접근 토큰을 받은 사용자는 자신이 보유한 키 UK_n와 GK_n를 이용하여, 새로운 Token Authenticator를 생성한다. 블록 접근 토큰 형식은 Table 2와 같으며, DataNode는 KeyNode로부터 사용자의 키값을 받아서 Token Authenticator를 검증한 후 블록을 사용자에게 전송한다.

4. 안전성 및 성능 분석

기존의 하둡 분산 파일 시스템은 파일에 접근할 때, 소유자에 대한 별도의 접근 권한을 확인하지 않아 공격자가 블록 접근 토큰을 취득하면 데이터를 제공받을 수 있으며, NameNode와 DataNode 사이에 공유된 비밀키값을 취득하면 모든 DataNode에 있는 블록에 접근할 수 있는 취약점을 가지고 있다. 또한 인가된 사용자일 경우 블록에 대해 읽기, 쓰기, 복사, 대체의 모든 권한을 구분 없이 가질 수 있는 취약점을 가지고 있다.

Table 3. Security Analysis

Security Items	Kerberos	PKI	Hash-chain	Group Key
Token Issued	Linux account	Individual	Linux Account	Individual and Group
Token Used (Replay Attack)	Possible	Impossible	Impossible	Impossible
Token Used (Impersonation Attack)	Possible	Impossible	Impossible	Impossible
Check Access Permissions	Impossible	Partially	Impossible	Possible

제안하는 시스템은 클라이언트의 그룹을 확인하고, 클라이언트의 그룹이 파일 소유자와 어떤 관계를 가지느냐에 따라 접근 권한을 제공할 수 있도록 접근 권한을 확인하였다. 블록 접근 토큰의 경우, 하나의 비밀키를 사용하여 접근하는 취약점을 해결하기 위해서 DataNode의 그룹키를 이용하여 블록 접근 토큰 키를 생성하였다. 또한 하나의 DataNode 그룹키만을 사용하는 것이 아닌, 블록이 저장된 DataNode들의 그룹키를 사용함으로써, 그룹키 하나가 유출되었다고 하더라도 다른 위치의 DataNode 그룹키를 알지 못한다면 해당 블록에 접근할 수 없다. 마지막으로, 클라이언트가 토큰 ID값과 자신의 비밀키를 이용하여 토큰 인증자값을 재생성하는 것이 아닌, 기존의 토큰 인증자값은 그대로 두고, 클라이언트의 비밀키를 추가하였으며, 토큰 ID에 있는 ownerID를 가지고 AS에게 비밀키를 요청하기 때문에, ownerID와 토큰 인증자값을 재생성한 클라이언트가 동일하여야 토큰 인증자를 인증할 수 있다.

Table 3은 하둡 분산 파일 시스템의 취약점에 대해 현재 사용하고 있는 Kerberos 기반 인증과 PKI 인증 기법, 해시 체인을 이용한 기법, 그리고 본 논문에서 제안하는 그룹키 기반의 인증 기법을 비교한 것이다.

토큰을 사용할 때 사용자 인증을 통해 발행된 블록 접근 토큰이 재사용되지 못하도록 해야 하는데, 기존의 Kerberos 인증에서는 사용자 인증을 하지 않고 있으며, PKI 기반이나 제안하는 그룹키 인증에서는 사용자가 보유한 인증서나 키값을 이용하여 사용자 인증을 하고 있다. 그러나 PKI 기반

의 경우 비대칭키 인증 방식으로 대칭키 인증 방식과 해시 연산에 비해 연산속도가 느리다는 단점이 있고, 해시 체인의 경우 NameNode가 사용자에게 토큰을 발행할 때 해시 연산을 n_x-1만큼 해야 하기 때문에 NameNode에 오버헤드가 발생할 수 있다는 문제점이 있다.

그룹키 인증방식은 NameNode나 KeyNode에서 발생할 수 있는 오버헤드를 방지하기 위해서 기존의 인증방식에 사용자가 자신이 보유하고 있는 키값을 이용하여 해시 연산을 하도록 하고 있으며, 블록 접근 토큰을 발행할 때 발생하는 연산 방식에 대해서 정리한 내용은 Table 4와 같다.

Table 4. Performance Analysis

Security Items		Kerberos	PKI	Hash-chain	Group Key
Algorithm		Symmetric + Hash	Asymmetric + Hash	Symmetric + Hash	Symmetric + Hash
The Number of Hash Algorithm	Name Node	1	-	(nx-1)+1	1
	Data Node	1	1	1	2
	Client	-	1	-	1

* n : Number of Clients
x : Number of Client Groups

Fig. 5, 6, 7은 보안이 적용되지 않은 하둡 시스템, PKI 기반 하둡 시스템, 커버로스 기반 하둡 시스템, 해시 체인 기반 하둡 시스템, 그리고 제안하는 시스템의 키 생성 속도를 비교한 그래프이다.

성능 분석을 위해서 정의한 알고리즘은 AES-CTR 256, RSA 1024, HMAC-SHA1이며, 성능 분석은 Microsoft Visual C++ 2005 SP1, Intel Core 2 1.83GHz, 윈도우 비스타(Vista) 32bit 환경에서 암호 알고리즘의 속도를 측정한 데이터[9]을 기반으로 작성하였다. 성능 분석 결과, 제안하는 시스템은 NameNode에서 커버로스 기반 하둡 시스템에 비해 평균 약 0.017ms의 키 생성 오버헤드를 보여주고 있으며, DataNode에서는 커버로스와 해시 체인 기반 하둡 시스템에 비해 약 0.0007ms의 키 생성 오버헤드만을 보이는 것으로 분석되었다.

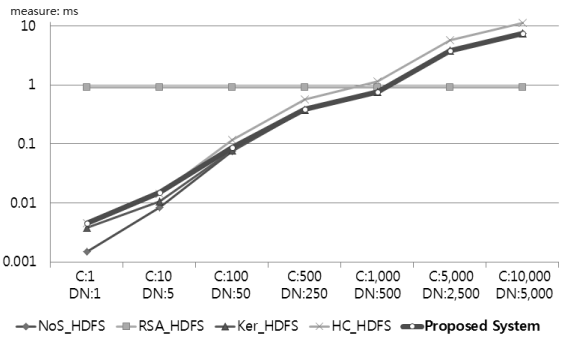


Fig. 5. Rate of Key Generation on NameNode

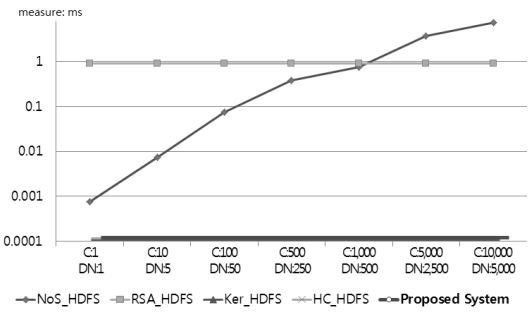


Fig. 6. Rate of Key Generation on DataNode

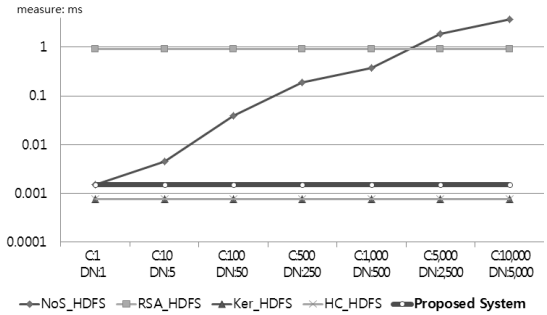


Fig. 7. Rate of Key Generation on Client

5. 결 론

본 논문에서는 하둡 분산 파일 시스템에서 블록 접근 토큰의 재사용 공격과 블록에 대한 접근 권한을 확인하지 않는 것에 취약점을 해결하고자 그룹키를 이용한 인증 방식을 제안하였다. 사용자가 가입할 때 그룹키를 생성하거나 변경해야 하기 때문에 오버헤드가 발생할 수 있으나, 사용자의 키값과 그룹키값을 이용하여 사용자 인증 및 블록에 접근 권한이 있는지 확인하고, 권한에 따라 별도의 접근 모드를 부여할 수 있었다. 또한 NameNode와 DataNode 사이에 공유된 비밀키값이 유출되더라도, 사용자가 보유하고 있는 키값들을 알지 못한다면, 블록에 접근할 수 없도록 하였다. 제안된 방식은 기존의 하둡 보안 방식들과 비교하였을 때, 큰 연산을 하지 않고도 안전성이 높은 것으로 밝혀졌다.

References

[1] O. G. Min, H. Y. Kim, and G. H. Nam, “Trends in Technology of Cloud Computing,” *Electronics and Telecommunications Trends*, Vol.24, No.4, pp.1-13, 2009.

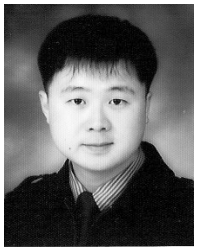
[2] A. Melnikov, et al., “Simple Authentication and Security Layer,” Internet RFC 4422, Jun., 2006.

[3] O. O’Malley, K. Zhang, S. Radia, R. Marti, and Ch. Harrell, “Hadoop Security Design,” 2009, Available From: https://issues.apache.org/jira/secure/attachment/12428537/security-design.pdf, accessed Feb., 28, 2014.

[4] Govind S, “Hadoop-Really a Preferred Approach over Relational Database Management Systems?,” Available From: https://www.academia.edu/3474885, accessed Feb., 17, 2014.
[5] S. Park, H. Kim, “Improving Hadoop Security Through Hash-chain,” *Journal of Korean institute of information technology*, Vol.10, No.6, pp.65-73, 2012.
[6] A. Becherer, “Hadoop Security Design Just Add Kerberos? Really?,” iSEC PARTNER, 2010.
[7] S. H. Park, I. R. 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.
[8] D. Borthakur, “The Hadoop Distributed File System: Architecture and Design,” The Apache Software Foundation, Available From: http://hadoop.apache.org/docs/r0.18.0/hdfs_design.pdf, 2007.
[9] Wei Dai, “Crypto++ 5.6.0 Benchmarks,” Available From: http://www.cryptopp.com/benchmarks.html, Mar., 2009.

김 형 주
e-mail : hyungjoo.kim@ssu.ac.kr
2008년 단국대학교 컴퓨터과학과(공학사)
2010년 숭실대학교 컴퓨터학과(공학석사)
2010년~현 재 숭실대학교 컴퓨터학과 박사과정
관심분야 : Authentication, Big-Data, Cloud Computing, IoT, Security

유 한 나
e-mail : belover7@naver.com
2010년 숭실대학교 컴퓨터학과(공학석사)
2014년 숭실대학교 컴퓨터학과(공학박사)
2014년~현 재 KT 융합연구소 연구원
관심분야 : Authentication, Big-Data, PKI



강 정 호
e-mail : kjh7548@naver.com
2000년 서울과학기술대학교 컴퓨터공학(공학사)
2002년 서울과학기술대학교 컴퓨터공학(공학석사)
2013년 숭실대학교 컴퓨터학과(공학박사)
관심분야 : Big-Data, NFC, Security

전 문 석
e-mail : mjun@ssu.ac.kr
1989년 Univ. of Maryland Computer Science (공학박사)
1991년~현 재 숭실대학교 컴퓨터학과 교수
관심분야 : RFID, PKI