Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.29, NO.6, Dec. 2019 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2019.29.6.1305

빅데이터 플랫폼 환경에서의 워크로드별 암호화 알고리즘 성능 분석*

이 선 주,[1†] 허 준 범[2‡]
[1]고려대학교 컴퓨터정보통신대학원 소프트웨어보안학과, [2]고려대학교 정보대학 컴퓨터학과

Analysis of Encryption Algorithm Performance by Workload in BigData Platform*

Sunju Lee,[1†] Junbeom Hur[2‡]
[1]Department of Software Security, Korea University Graduate School of Computer & Information Technology,
[2]Department of Computer Science and Engineering, Korea University College of Informatics

요 약

공공기관 및 기업의 빅데이터 플랫폼 환경에서 데이터 보호를 위한 암호화는 필수적인 요소이나 실제 빅데이터 워크로드를 고려한 암호화 알고리즘에 대한 성능 검증 연구는 많이 진행되지 않았다. 본 논문에서는 몽고DB(MongoDB) 환경에서 데이터와 노드를 추가하여 빅데이터의 6가지 워크로드별로 AES, ARIA, 3DES별로 성능 변화 추이를 분석하였다. 이를 통해 빅데이터 플랫폼 환경에서 각 워크로드 별 최적의 블록기반 암호 알고리즘이 무엇인지 확인하고, NoSQL 데이터베이스 벤치마크(YCSB)를 사용하여 데이터와 노드 구성환경에서 다양한 워크로드별로 테스트를 통해 MongoDB의 성능을 고려한 최적화된 아키텍처를 제안한다.

ABSTRACT

Although encryption for data protection is essential in the big data platform environment of public institutions and corporations, much performance verification studies on encryption algorithms considering actual big data workloads have not been conducted. In this paper, we analyzed the performance change of AES, ARIA, and 3DES for each of six workloads of big data by adding data and nodes in MongoDB environment. This enables us to identify the optimal block-based cryptographic algorithm for each workload in the big data platform environment, and test the performance of MongoDB by testing various workloads in data and node configurations using the NoSQL Database Benchmark (YCSB). We propose an optimized architecture that takes into account.

Keywords: MongoDB, Encryption, BigData, NoSQL, YCSB

Received(11. 11. 2019), Modified(12. 12. 2019), Accepted(12. 16. 2019)
* 본 논문은 2019년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (No.2019-0-00533, 컴퓨터 프로세서의 구조적 보안 취약점 검증 및 공격 탐지대응)
† 주저자, uamybest@korea.ac.kr
‡ 교신저자, jbhur@korea.ac.kr(Corresponding author)

I. 서 론

1.1 연구의 배경

암호모듈 검증은 전자정부법 시행령 제 69조와 〔암호모듈 시험 및 검증지침〕에 의거, 국가·공공기관 정보통신망에서 소통되는 자료 중에서 비밀로 분류되지 않은 중요 정보의 보호를 위해 사용되는 암호모듈의 안전성과 구현 적합성을 검증하는 제도이다. 검증이 필요한 주요 제품군으로는 DB 암호화, 통합인증(SSO), 문서 암호화(DRM 등, 메일 암호화, 구간 암호화, 디스크·파일 암호화, 하드웨어 보안 토큰과 같이 암호가 주기능인 정보보호 시스템은 필수적으로 암호 모듈에 대해 검증하도록 지정되어 있다.

검증대상 암호 알고리즘은 안전성·신뢰성·상호운용성 등이 적합한 국내·외 표준 암호 알고리즘으로서 블록암호(ARIA, SEED, LEA, HIGHT), 해시함수(SHA-2, LSH, SHA-3), 메시지인증(HMAC, CMAC, GMAC), 공개키 암호(RSAES), 전사서명(RSA-PSS, KCDSA), 전자서명, 키설정 등이 있다〔1〕.

암호모듈 검증 시험·평가 기관은 2010년 이래로 72개 사 183건의 검증을 하였으며, 필 제품을 공개함으로써 「개인정보보호법」 및 「정보통신망 이용촉진 및 정보보호 등에 관한 법률」에 의거 민감한 개인정보를 암호화하고자 하는 공공기관과 기업에서 검증한 제품을 도입하도록 권고하고 있다〔2〕.

그런데,「개인정보보호법」 및 「정보통신망 이용촉진 및 정보보호 등에 관한 법률」에 의거 암호화를 수행한 공공기관과 기업 78개사를 대상으로 암호화 시 선정한 알고리즘 현황을 조사한 결과 약 3%만이 국내에서 개발한 ARIA 알고리즘을 선택한 것으로 나타났다. 대부분의 공공기관과 기업은 개인정보 암호화 시 AES 알고리즘으로 암호화를 수행하였다. 또한, 암호 알고리즘 선정 시 가장 크게 고려하는 요소로 성능 및 기존 시스템 영향도를 가장 크게 들었다.

최근 정부 관련 부처와 국회에서는 빅데이터 활용을 통해 관련 산업 활성화를 위한 규제를 완화하기 위한 법규(빅데이터의 이용 및 산업진흥 등에 관한 법률)를 제정하기 위한 논의가 진행되고 있다〔3〕〔4〕.

1.2 연구의 목적

기업과 공공기관에서는 빅데이터 시스템 도입이 점차 증가하고 있으며, 다양하고 방대한 데이터를 저장하는 빅데이터 시스템 또한 개인의 민감한 정보에 대해 암호화 이슈는 동일하게 제기될 수 있다. 빅데이터 플랫폼 환경에서 암호 알고리즘별로 데이터를 증가에 따라 다양한 워크로드에 대해 암호화 성능이 어떻게 변화되는지 검증하고자 한다.

또한, 데이터 저장을 위한 몽고DB 환경에서 데이터 분산처리 성능향상을 위한 Replica Set을 추가했을 때 워크로드별로 암호화 성능분석을 통해 최적화된 아키텍처를 제시하고자 한다.

1.3 논문의 구성

본 논문의 구성은 다음과 같다. Ⅱ장은 빅데이터 플랫폼 환경에서의 암호화 성능 관련 연구들을 소개한다. Ⅲ장과 Ⅳ장은 빅데이터 플랫폼(MongoDB)의 구조 및 플랫폼에서 제공하는 데이터 보안의 방법을 소개한다. Ⅴ장은 테스트 환경 및 데이터 구성, 테스트 방법을 소개하고, Ⅵ장에서 수행한 테스트 결과 및 분석에 대해 데이터 건수와 노드를 증가시켰을 때 성능이 변화 추이에 대해 제시한다. 마지막으로 Ⅵ장에서는 결론에 관해 기술한다.

II. 관련 연구 및 암호 알고리즘

2.1 빅데이터 플랫폼 암호화 관련 연구

몽고DB는 NoSQL로 분류되는 크로스 플랫폼 도큐먼스 지향 데이터베이스 시스템이다. 10gen에서 2007년 10월 계획된 PaaS(Platform as a Service) 제품의 구성 요소로 처음 개발하였으며 10gen이 상용 지원 및 기타 서비스를 제공한 2009년에 오픈 소스 개발 모델로 전향하였다. 그 뒤로 몽고DB는 크레이그리스트, 이베이, 포스퀘어, 소스포지, 뉴욕 타임즈, 구글, 페이스북와 같은 수많은 주요 웹사이트 및 서비스에 백엔드 소프트웨어로 채택되고 있다. 몽고DB는 가장 유명한 NoSQL 데이터베이스 시스템이다〔5〕.

데이터 저장을 위해 관계형 데이터베이스를 이용하여 모든 워크로드를 처리하였지만, 빅데이터 플랫폼과 같이 대량의 데이터 처리에는 만족할 만한 성능

을 내지 못하고 있다. 몽고DB를 사용하는 빅데이터 시스템 도입이 늘어나면서 부족하지만, 관련 연구가 발표되고 있다[6]. 기존 사용자들이 관계형 데이터베이스 익숙한 사용자들의 관점에서 NoSQL 데이터베이스로 시스템을 구축할 때 발생할 수 있는 성능 차이를 중심으로 연구가 진행되었다[7][8]. 빅데이터 시스템 도입 시 데이터 보안을 강화하기 위해 암호화를 적용하는 경우 알고리즘에 대해 다양한 워크로드별 성능 변화 추이에 관한 연구는 미미한 실정이다[9].

2.2 빅데이터 암호화 알고리즘 성능 분석

몽고DB는 데이터 저장 시 암호화 방식을 2016년도 제공하고 있으며, 지원하는 알고리즘도 AES로 한정되어 있다. 그래서 다양한 암호 알고리즘을 활용한 암호화 적용 방법에 관한 연구뿐만 아니라 적용 시 성능 변화에 관한 연구가 미미한 상태이다. 반면 맵리듀스를 구현한 대표적인 빅데이터 분산 처리 시스템인 하둡의 경우 AES, ARIA 등 다양한 알고리즘을 활용한 암호화 사례뿐만 아니라 성능에 관한 연구도 활발하게 진행되고 있다[10][11].

2.3 암호화 알고리즘

2.3.1 AES 알고리즘 분석

AES(Advanced Encryption Standard)는 Rijmen과 Daemon이 개발한 암호 알고리즘으로 암호화 블록 크기의 크기는 128비트이며 암호화 키의 길이가 128, 192, 256비트를 사용하는 알고리즘이다. AES 알고리즘의 암·복호화 과정은 Add Round Key , Sub Byte, Shift Row, Mix Column 이 반복 되어 이루어진다[12].

- Add round key : 데이터와 키를 XOR한다.
- Sub Type : 8비트 단위로 데이터를 치환하는 연산으로서, 실제 연산은 1바이트를 GF(2^8) 상에서의 역원을 구한 후 연산이다.
- Shift row : 행단위로 Shift 연산을 한다.
- Mix column : 각각의 바이트에 특정행렬과 곱연산을 가하여 변환한다.

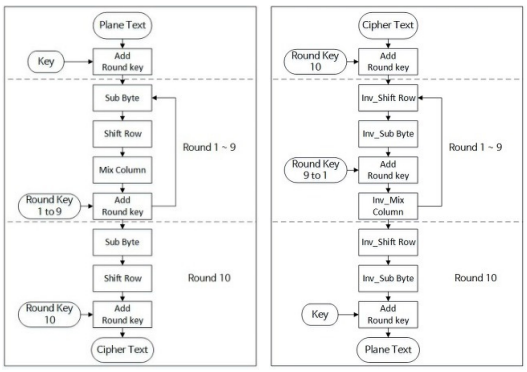

Fig. 1. AES En/Decryption Flow

2.3.2 3DES 알고리즘 분석

3DES(Triple Data Encryption Algorithm)는 DES를 세 번 연이어 암호화–복호화–암호화 연산하는 암호 알고리즘으로 1998년 배포되었다. DES는 64bit의 평문을 56bit의 키를 사용하여 F함수 연산과 32bit씩 좌우로 나누어 교차하는 과정을 16라운드에 걸쳐 수행하는 알고리즘이다[13].

- Initial Permutation : IP표에 의해서 평문을 치환하고 라운드 함수가 동작하며, 라운드 함수가 끝나면 IP^{-1}로 역연산하게 된다.
- Sub Key Generator : 생성된 각각의 서브키와 원래의 키는 서로
- F 함수 : 치환과 대체를 수행하는 함수
- XOR 연산 : 역연산이 가능해 복호화하는 알고리즘을 역수행 한다.

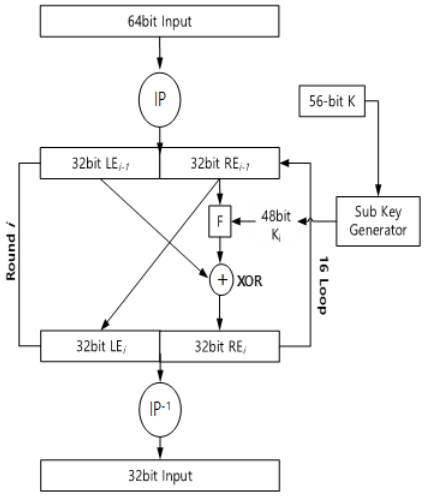

Fig. 2. DES Encryption Flow

2.3.3 ARIA 알고리즘 분석

ARIA(Academy, Research Institute, Agency)는 국가보안기술연구소 주도로 개발한 국내 암호화 알고리즘으로 우리나라 전자정부 안정성 강화를 목적으로 개발되었다. AES와 동일한 인터페이스를 갖도록 설계되어 있다[14].

- Substitution Layer : 두 유형의 치환 계층(S-box Layer Type)이 있으며, 각각은 2종의 8비트 입·출력 S-box와 그들의 역변환으로 구성된다.
- Diffusion Layer : 16x16 Involution 이진 행렬을 사용한 바이트 간의 확산 함수로 구성되어있다.
- Key Expansion, AddRoundKey : 최화 과정에서는 암·복호화 한 라운드를 F함수로 하는 256비트 입·출력 3라운드 Feistel 암호를 이용한다.

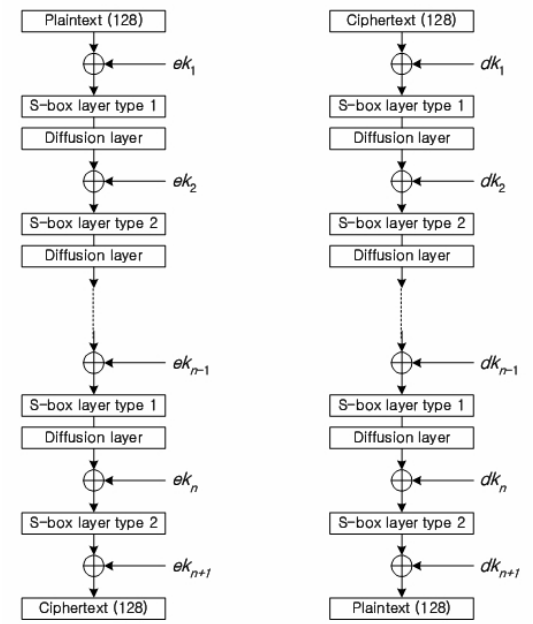

Fig. 3. ARIA Enc/Decyption Flow

III. 빅데이터 플랫폼(MongoDB) 구조

3.1 NoSQL DBMS

일반적으로 DB는 데이터를 열과 오를 맞추어 테이블이란 곳에 저장해 놓고 쉽게 찾을 수 있는 것을 말한다. 이러한 DB의 종류로는 관례형 DB, 그래프 DB, NoSQL DB 등 여러 가지 DB가 있다. 전통적인 관계형 DBMS는 빅데이터 환경과 같이 극단적으로 데이터가 크고 READ, WRITE 간의 격차가 큰 경우에는 적합하지 않은 문제가 있어 NoSQL(Not Only SQL)이 출현하게 되었다.

NoSQL DBMS는 여러 서버에 데이터를 나누어 저장하고, 데이터 모델을 단순화해서 분산을 위한 샤딩(sharding) 기술을 접목하게 되었다. 또한, 분산 복제 환경에서 일관성의 요건을 완화하거나, 격리 요건을 제약하는 형태로 발전하게 되었다. 2018년도 NoSQL 데이터베이스 활용 순위를 보여주는 Table 1.과 같이 몽고DB는 NoSQL DB 분야에서 가장 많이 활용되고 있다[15].

Table 1. NoSQL Database Usage Rankings in 2019

Rank			DBMS	Database Model	Score		
Oct. 2019	Sep. 2019	Oct. 2018			Oct. 2019	Sep. 2019	Oct. 2018
1	1	1	Oracle detailed information	Relational, Multi-model	1355.88	+9.22	+36.61
2	2	2	MySQL detailed information	Relational, Multi-model	1283.06	+3.99	+104.94
3	3	3	Microsoft SQL Server detailed information	Relational, Multi-model	1094.72	+9.66	+36.39
4	4	4	PostgreSQL detailed information	Relational, Multi-model	483.91	+1.66	+64.52
5	5	5	MongoDB detailed information	Document, Multi-model	412.09	+2.03	+48.9
6	6	6	IBM Db2 detailed information	Relational, Multi-model	170.77	-0.79	-8.91
7	7	8	Elasticsearch detailed information	Search engine, Multi-model	150.17	+0.9	+7.85
8	8	7	Redis detailed information	Key-value, Multi-model	142.91	+1.01	-2.38
9	9	9	Microsoft Access	Relational	131.18	-1.53	-5.62
10	10	10	Cassandra detailed information	Wide column	123.22	-0.18	-0.17

3.2 MongoDB 아키텍처

몽고DB는 Fig.1과 같은 아키텍처로 구성되어 있다. mongos는 애플리케이션의 요청에 대해 각각의 샤드 서버들의 연결을 해주는 라우터 역할(Load

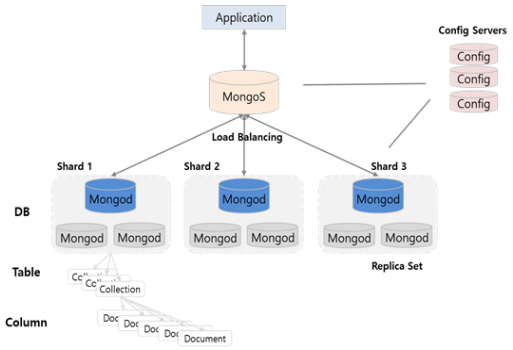

Fig. 4. MongoDB Architecture Structure

Balancing)을 수행하는 구조로 되어 있다.

몽고DB의 아키텍처를 기존의 RDBMS와 비교하여 설명하면, 샤드는 데이터를 저장하는 데이터베이스이며, 컬렉션(collection)은 테이블, 도큐먼트는 컬럼이라고 표현할 수 있다. Fig.1.에서 표현된 구성 및 역할에 관해 설명하면 다음과 같다[16].

- mongos는 어플리케이션의 요청을 받아 컨피그 서버의 데이터 위치 정보를 참고해 적절한 데이터(mongod) 서버로 요청을 포워딩
- mongod는 데이터를 저장, 관리하는 서버
- 컨피그 서버는 샤딩에 대한 환경 설정 정보 및 데이터 위치 정보를 관리
- 리플리카 셋(=샤드)은 이중화를 구현하기 위한 기술로서 하나의 주 노드와 2개 이상의 부 노드로 구성되며, 데이터를 실시간으로 공유함으로써 높은 가용성을 지원하는 기술

IV. MongoDB 데이터 보안

4.1 네트워크 구간 암호화

몽고DB는 네트워크 구간 암호화를 위해 어플리케이션 드라이버를 통해 데이터 암호화 및 TLS/SSL을 이용하여 네트워크 구간 암호화 지원(OpenSSL Library를 이용)한다. 또한, 쿼리 라우터(mongod)와 샤드(mongos) 및 Node 간에 네트워크 연결 시 트래픽 보호를 위해 SSL/TLS 1.1+를 이용하여 중요 데이터에 대한 보안을 한층 강화할 수 있도록 지원하고 있다.

Fig.5. 는 네트워크 구간 암호화를 보여주는 그림으로써 애플리케이션(사용자)의 요청에 대해 TLS/SSL 드라이버를 이용하여 쿼리 라우터와 각 리플리카 셋 간의 네트워트 전송 시 TLS/SSL를 적용하여 보완 할 수 있다[17].

몽고DB의 TLS/SSL 연결방식에는 4가지가 있으며, 환경에 맞게 선택적으로 적용할 수 있게 되어 있다.

- Disabled : TLS/SSL 사용하지 않음
- allowSSL : 서버 간에는 TLS/SSL 사용하지 않고, 들어오는 연결에 대해서 TLS/SS 연결이든 아니든 모두 허용
- preferSSL : 서버 간의 연결은 TLS/SSL 사용하고, 들어오는 연결에 대해서는 TLS/SS 연결이든 아니든 모두 허용
- requireSSL : TLS/SSL 암호화 연결만 사용

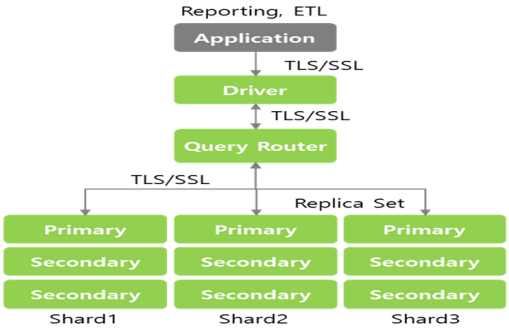

Fig. 5. Network Encryption

4.2 데이터 저장 시 암호화

몽고DB의 데이터 저장기술인 와이어드타이거 스토리지 엔진기반의 데이터 암호화를 제공하고 있다. 현재 지원하는 암호 알고리즘은 AES256-CBC, AES256-GCM 암호화 방식만 제공하고 있다. 암호화를 위해 2가지 형식의 Key를 사용하고 있는데, 각각의 리플리케이션 셋을 암호화하기 위한 마스터 키와 데이터베이스를 암호화할 때 사용하는 내부 키를 사용하고 있다[18].

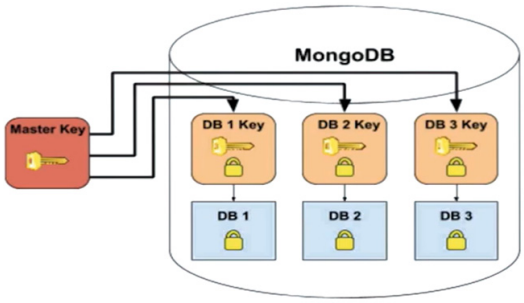

Fig. 6. Date Encryption at rest

몽고DB에서 데이터를 암호화하는 과정은 먼저 마스터 키를 생성한 후 각 데이터베이스를 암호화할 때 사용하기 위한 내부 키를 생성하게 된다. 이후 내부 키를 이용하여 데이터를 암호화하여 저장하게 되고 내부 키를 보호하기 위해 마스터 키로 한 번 더 암호화를 하는 방식으로 진행하게 된다.

V. 암호화 적용 및 성능 테스트 환경

5.1 테스트 환경

암호화 적용을 위한 테스트 환경은 빅데이터 플랫폼 환경에서 일반적으로 사용하는 오픈소스를 기반으로 구성을 하였다. 테스트 환경은 하드웨어 용량을 고려하여 Fig.7.과 같이 구성하였으며, 세부적인 테스트환경 구성은은 다음과 같이 하였다.

- 운영체제 : CentOS v7.0(CPU 4, 메모리 8GB, Disk 100GB)
- MongoDB : v3.6.14
- Replica Set : 3-Node, 6-Node 데이터 복제 (Primary-Secondary-Secondary)
- 암호 알고리즘 : AES256, 3DES, ARIA256
- Key Management System : Thales v6.0

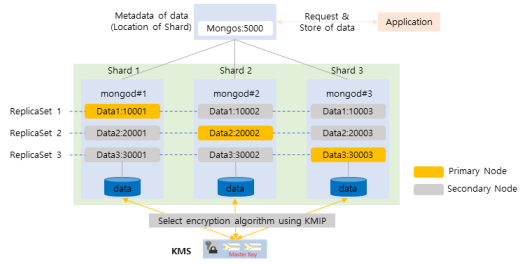

Fig. 7. Encryption Test Environment

5.2 테스트 데이터 셋

빅데이터 시스템은 기존의 레거시시스템과 다른 많은 데이터를 저장하고 있는 시스템이다. 암호 알고리즘별로 테스트를 했을 때 성능에 대한 변별력을 확보하기 위해서는 데이터양을 중요한 요소로 고려하였다. 빅데이터를 생성해 주는 다양한 데이터 생성 도구가 있지만, 시간이 많이 필요한 점을 고려하여 빅데이터 활성화를 위해 공공데이터 포털사이트에서 제공해 주는 데이터 셋을 활용하였다.

Fig. 8. Test Data Set

공공데이터 포털사이트에서 제공하는 정보 중 국민건강보험공단이 제공하는 2016년도의 건강검진 정보 데이터를 내려받아 약 500만 건으로 재가공하여 몽고DB에 load 하여 구성하였다[19].

5.3 알고리즘별 암호화 적용

'4.2 데이터 저장 시 암호화'에서 언급한 바와 같이 몽고DB는 데이터 암호화를 위해 내부 키를 이용하여 데이터를 암호화하게 되고, 이때 사용한 내부 키와 데이터를 마스터 키를 이용하여 한 번 더 암호화를 하게 된다. 그런데, 암호화하는 과정에서 가장 중요한 암호 키를 서버 내부에 보관하는 점은 키 유출 위험이 크다고 할 수 있다.

이러한 문제점을 개선하기 위해 암호 키를 생성하고 관리하는 전용 키 관리시스템(KMS : Key Management System)을 구성하여 테스트하였다. Mongo 서버와의 통신은 암호 키 호출을 위한 표준 프로토콜인 KMIP(Key Management Interoperability Protocol)를 이용하여 암·복호화가 이루어지도록 구성하였다[20].

데이터 암호화는 KMS에서 제공하는 암호 알고리즘(AES, 3DES, ARIA)에 대해 각각의 UUID(Universally Unique Identifier)를 생성하여 적용하였다. mongod 프로세스는 데이터 입·출력이 발생하는 경우 UUID 호출을 통해 암·복호화가 수행되는 구조이다. 즉, 암호화하고자 하는 알고리즘의 UUID 선택 후 mongod를 기동하면 쓰기 또는 읽기가 수행되는 동안 데이터에 대해 암·복호화가 적용된다.

AES-256, 3DES, ARIA-256 알고리즘에 대한 UUID는 다음과 같다.

- AES:89d6d529-a2fe-31dd-ae60-3cb35458dcc0
- ARIA:da6fd698-4681-3bb5-88ad-d1d9d68b1df6
- 3DES:93a7f87f-458c-31be-ad16-d5dfa2fc1bae

Mongod 기동 시 사용하는 옵션을 이용하여 암호 알고리즘을 선택할 수 있는데, 옵션은 다음과 같다[21].

- enableEncryption : mongod의 암호화 기능

을 활성화

- kmipServerName : mongod와 전용 키관리 서버 간 통신을 위한 IP
- kmipServerCAFile : 키관리서버 연결을 위한 서버인증서 파일
- kmipClientCertificateFile : 클라이언트 인증을 위한 인증서 파일
- kmipKeyIdentifier : 암호 알고리즘별 UUID 값

5.4 워크로드 및 성능측정 방법

5.4.1 워크로드 테스트

빅데이터 환경에서 YCSB(Yahoo Cloud Serving Benchmark)를 이용하여 성능 테스트를 하는 경우 Table 2와 같이 6가지 형태의 서로 다른 워크로드를 적용하면서 성능의 변화를 측정할 수 있다[22][23].

Table 2. Workload Type of YCSB

Workload Scenario	Operations	Record Selection
A - Update Heavy	Read : 50%, Update : 50%	Zipfian
B – Read Heavy	Read : 95%, Update : 5%	Zipfian
C – Read Only	Read : 100%	Zipfian
D – Read latest	Read : 95%, Insert : 5%	Latest
E – Short ranges	Scan : 95%, Inserts : 5%	Zipfian / Uniform
F – Read-Modify-write	-	Zipfian

각각의 워크로드 적용 시 레코드 선택의 기준은 Zipfian, Latest, Uniform 알고리즘을 선정하였다[24].

- Uniform : 임의로 레코드를 선정한다.
- Latest : 가장 최근에 삽입된 레코드들을 선정
- Zipfian : 지프의 법칙을 적용하여 일부의 레코드만 선택확률을 높여 선정

YCSB를 이용하여 측정할 수 있는 항목은 Table 3.과 같으며, 본 성능 테스트에서는 처리량(Throughput) 값을 수집하여 결과값을 비교하였다[25].

Table 3. Measurement Criteria

Measurement Value	Unit	Definition
Throughput	ops	Unit throughput per second processed by unit instance
Av. Insert Latency	us	Average of insert latency for a instance
Av. Read Latency	us	Average read latency for instances
Av. Update Latency	us	Average update latency for instances
Av. Scan Latency	us	Average of range search latency for instances

5.4.2 성능측정 방법

성능측정은 두 가지 방법으로 나누어서 진행하였다. 첫 번째는 데이터를 100만 건, 300만 건, 500만 건으로 증가시키면서 각 데이터 건수에 대해 알고리즘별로 3회씩 측정하여 평균값으로 하였다. 두 번째는 노드 수를 3-Replica Set을 6-Replica Set으로 증가시킨 후 100만 건, 300만 건, 500만 건에 대해 알고리즘별로 3회씩 측정하여 평균값으로 하였다.

6가지 워크로드에 대해 100명의 동시 사용자가 100,000건의 오퍼레이션을 수행했을 때 초당 처리 트랜잭션의 평균값을 구하여 알고리즘별 성능 차이를 비교하는 방식으로 진행하였다.

VI. 연구 결과 및 분석

6.1 워크로드별 암호화 성능

다음으로 6가지 워크로드를 몽고DB에 적용하여 수행한 결과를 살펴보도록 한다. 100만, 300만, 500만 건의 데이터를 저장한 후 각 워크로드를 수행하였다.

워크로드 A는 Update Heavy가 발생하는 경우로서 Read 50%, Update 50% 비율로 부하를 발생시킨 결과이다.

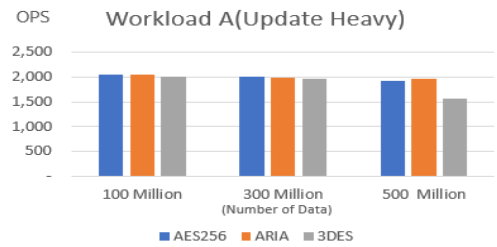

Fig. 9. Update Heavy Measurement Results

몽고DB에 100만 건, 300만 건의 데이터가 저장되어 있는 상황에서 알고리즘별로 업데이트 테스트를 했을 때, OPS(Operation Per Second)는 약 2,000회로서 알고리즘 간 차이가 없었다. 데이터를 500만 건으로 증가시켰을 때 AES와 ARIA는 비슷한 성능이 나왔지만, 3DES의 경우 100만 건으로 테스트했을 때 보다 23% 성능저하가 발생했다. 데

이터가 지속적으로 증가하는 시스템의 경우 컬렉션 데이터의 업데이트가 빈번하게 일어나는 경우가 많다. 업데이트가 많은 업무시스템인 경우 AES 또는 ARIA 알고리즘으로 암호화하는 것이 성능저하를 최소화 할 수 있을 것으로 판단된다.

워크로드 B는 Read Heavy가 발생하는 경우로서 Read 95%, Update 5% 조건으로 부하를 발생시킨 결과이다.

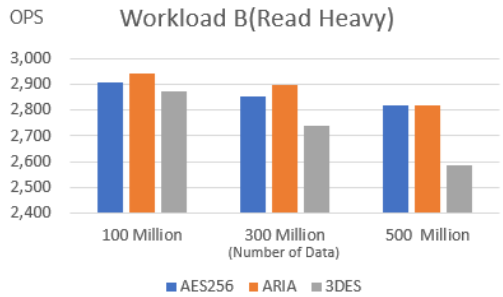

Fig. 10. Read Heavy Measurement Results

100만 건, 300만 건의 데이터가 저장되어 있는 상황에서 알고리즘별로 Read 테스트를 했을 때, 3개 알고리즘별로 미미한 성능 차이가 있었다. Read 중심의 업무에서는 ARIA 알고리즘이 가장 좋은 성능을 보였으며, 데이터를 500만 건으로 증가시켰을 때에도 안정적인 성능 변화를 나타냈다. AES의 경우 데이터 증가에 다른 성능 변화가 심했으며, 3DES의 경우 100만 건으로 테스트했을 때 보다 10% 성능저하가 발생했다. 컬렉션 내의 데이터 특성이 읽기 업무가 대부분인 경우 ARIA 알고리즘이 데이터 증가에 따라 안정적인 성능을 보여주는 것으로 나왔다.

워크로드 C는 Read Only가 발생하는 경우로서 Read 100% 조건으로 부하를 발생시킨 결과이다.

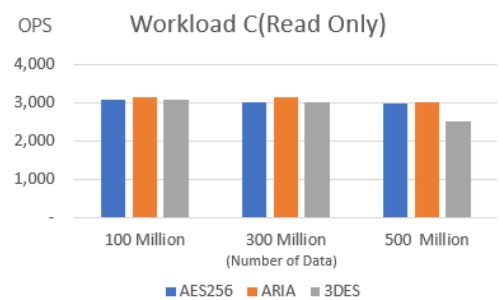

Fig. 11. Read Only Measurement Results

컬렉션의 데이터에 대해 읽기만 하는 경우경우에는 데이터 증가와 상관없이 3개 알고리즘 간 성능 차이는 크지 않았다.

워크로드 D는 Read latest가 발생하는 경우로서 Read 95%, Insert 5% 조건으로 부하를 발생시킨 결과이다.

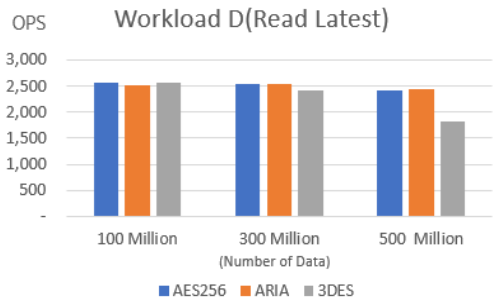

Fig. 12. Read Latest Measurement Results

몽고DB에 100만 건, 300만 건의 데이터가 저장되어 있는 상황에서 알고리즘별로 최근 Insert 되는 데이터에 대해 읽기 테스트를 했을 때, OPS는 약 2,500회로서 알고리즘 간 차이가 없었다. 데이터를 500만 건으로 증가시켰을 때 AES와 ARIA는 는 비슷한 성능이 나왔지만, 3DES의 경우 100만 건으로 테스트했을 때 보다 29% 성능저하가 발생했다. 컬렉션에 삽입하는 데이터의 경우 AES 또는 ARIA 알고리즘으로 암호화하는 것이 성능저하의 위험이 적을 것으로 판단한다.

워크로드 E는 Short Range Scan이 발생하는 경우로서 Scan 95%, Insert 5% 조건으로 부하를 발생시킨 결과이다.

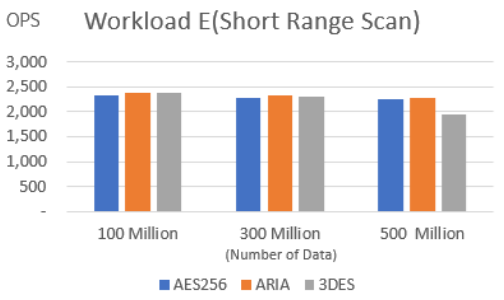

Fig. 13. Short Range Scan Measurement Results

컬렉션의 데이터에 대해 Full Scan이 아닌 짧은 범위의 검색 업무의 경우 3DES의 성능이 가장 안 좋게 나왔지만 큰 차이는 없었다. 3DES 알고리즘의 경우 워크로드 A~D에 대해 테스트 했을 때 보다 데이터 증가에 따른 성능 변화가 크지 않았다.

워크로드 F는 Read-Modify-Write가 발생하는 경우로서 저장된 데이터를 읽고 변경 후 다시 저장하는 조건으로 부하를 발생시킨 결과이다.

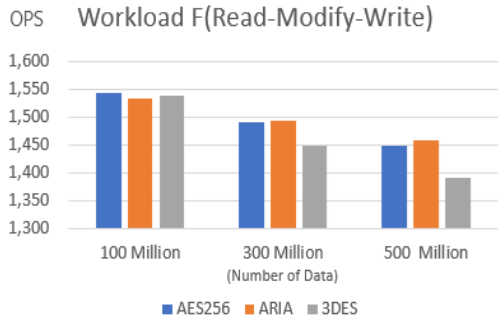

Fig. 14. Read Modify-Write Measurement Results

컬렉션의 도큐먼트를 읽기하여 데이터를 변경한 후 저장하는 업무 형태에 대해 테스트했을 때 데이터 증가에 따른 성능감소가 가장 많이 발생했다.100건의 데이터에 대하여 3개 알고리즘별로 테스트 결과 OPS는 1,500회 정도로 다른 워크로드에 비해 성능이 낮게 나왔다. 이는 3가지 오퍼레이션을 복합적으로 구성하여 워크로드를 주었기 때문에 발생한 현상이다. 데이터를 300만 건으로 증가시켰을 때 100만 건보다 3~6% 정도 성능저하가 발생했지만 3개 알고리즘 간에 차이는 미미했다. 하지만 데이터를 500만 건으로 증가시켰을 때 AES와 ARIA는 비슷한 성능이 나왔지만, 3DES의 경우 1,391 OPS로 100만 건으로 테스트했을 때 보다 Read Latest의 경우 10% 성능저하가 발생했다. 해당 데이터는 AES와 ARIA의 성능저하 수치인 5~6%보다 훨씬 더 많이 성능저하가 발생함을 확인할 수 있었다. 업무시스템 중 기존 데이터를 수정하는 업무가 많은 경우 성능 변동 폭이 크므로, 컬렉션 생성 및 배치 시 물리적 설계를 고려할 필요가 있다.

6.2 노드 수에 따른 암호화 성능

몽고DB의 데이터 저장 Node를 3-ReplicaSet에서 6-ReplicaSet으로 증가시킨 후 데이터 건수별로 6가지 워크로드에 대해 알고리즘별 성능 테스트를 수행한 결과이다.

동일한 환경에서 트랜잭션 분산처리 성능을 향상을 위해 리플리카 셋을 추가한 후 알고리즘별 성능 변화 추이를 분석하였다. 몽고DB를 활용하여 빅데이터 플랫폼을 구축하는 경우 최적의 아키텍처를 검증하였다.

각 노드에 대해 100만, 300만, 500만 건의 데이터를 각각 저장한 후 알고리즘별로 각각 3회씩 테스트를 수행하여 데이터 건수별로 성능 테스트를 수행한 결과에 대한 평균값을 각 워크로드에 대한 측정값으로 하였다. 각 워크로드마다 100명의 동시사용자가 100,000개의 오퍼레이션을 수행하도록 했다.

워크로드 A는 Update Heavy가 발생하는 경우로서 Read 50%, Update 50% 조건으로 부하를 발생시킨 결과이다.

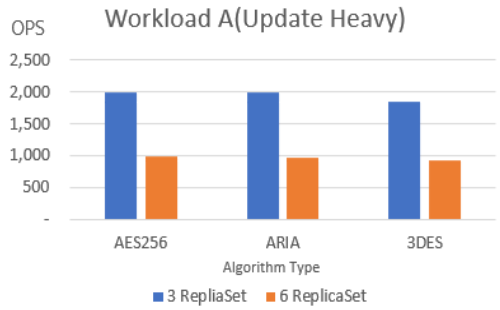

Fig. 15. Update Heavy Measurement Results

DB에 100만 건, 300만 건의 데이터가 저장되어 있는 상태에서 알고리즘별로 업데이트 테스트를 했을 때, 3-ReplicaSet에 비해 6-ReplicaSet에서는 OPS가 49~50% 정도로 성능저하가 발생했다. 데이터 증가에 따른 알고리즘 간 성능 차이가 없었지만, 트랜잭션 동시처리 속도가 향상될 것으로 예상했던 리클리카 셋 추가는 오히려 성능이 현저히 떨어지는 것으로 나왔다.

데이터를 500만 건으로 증가시킨 후 업데인트가 많은 업무 또는 컬렉션에 대해 성능 테스트를 수행한 결과 3DES의 경우 22%의 성능저하가 발생했다.

워크로드 B는 Read Heavy가 발생하는 경우로서 Read 95%, Update 5% 조건으로 부하를 발생시킨 결과이다.

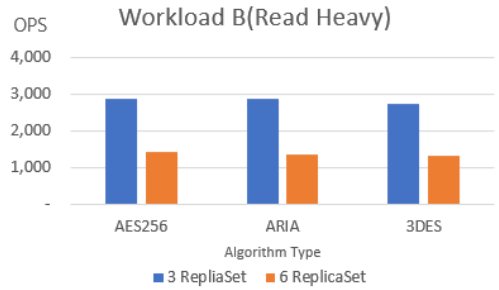

Fig.16. Read Heavy Measurement Results

읽기 중심의 업무에 대해 알고리즘별로 성능 테스트를 수행한 결과 3-ReplicaSet에 비해 6-ReplicaSet에서는 OPS가 47~49% 정도의 성능저하가 발생했다. 6-ReplicaSet에서 데이터를 증가해서 측정한 결과 AES와 ARIA는 4~6% 성능저하가 발생했지만, 500만 건에서는 3DES의 경우 성능저하가 가속되었다.

워크로드 C는 Read Only가 발생하는 경우로서 Read 100% 조건으로 부하를 발생시킨 결과이다.

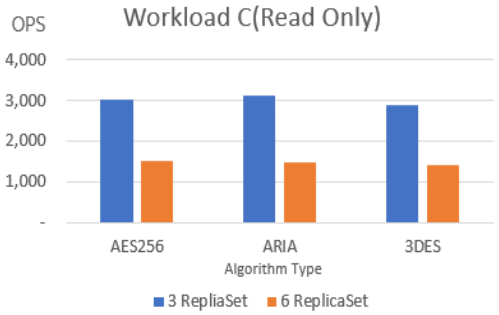

Fig. 17. Read Only Measurement Results

데이터가 저장되어 있는 컬렉션에 대해 Read만 수행하는 업무에 대해 테스트를 수행하였다. 리플리카 셋 추가에 따른 성능저하 외에 알고리즘 간 성능 차이는 미미했다.

워크로드 D는 Read latest가 발생하는 경우로서 Read 95%, Insert 5% 조건으로 부하를 발생시킨 결과이다.

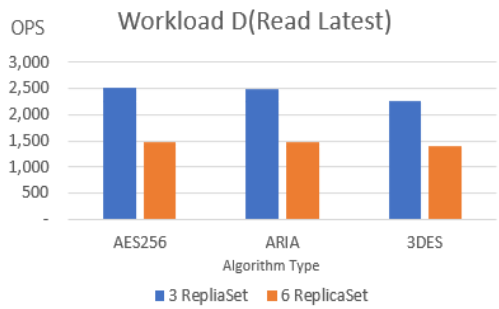

Fig. 18 Read Latest Measurement Results

가장 최근에 업데이트 된 데이터가 가장 많이 활용될 것으로 예상되는 업무를 중심으로 테스트를 수행한 결과 Read Heavy, Read Only에 비해 OPS 성능은 약 25% 정도 성능이 떨어졌다. 데이터 증가에 따라 AES와 ARIA 알고리즘에서의 Read Latest 테스트 성능 차이는 없었다. 하지만, 3DES의 경우 6-ReplicaSet으로 증가시킨 후 테스트 한 결과 3-ReplicaSet에 비해 62% 정도의 성능저하가 발생했다.

워크로드 E는 Short Range Scan이 발생하는 경우로서 Scan 95%, Insert 5% 조건으로 부하를 발생시킨 결과이다.

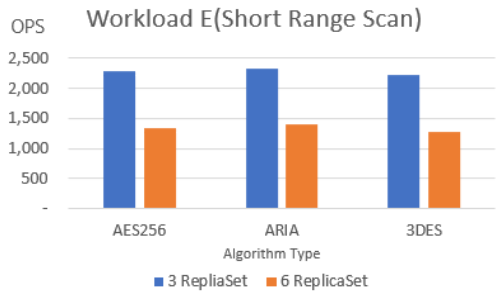

Fig. 19. Short Range Scan Measurement Results

컬렉션의 데이터에 대해 전체 검색이 아닌 짧은 범위의 검색 업무의 경우 알고리즘간 성능 차이는 크게 없었다.

6-ReplicaSet에서 데이터를 증가해서 측정한 결과 3개의 알고리즘 간 성능 차이는 3~7% 성능저하가 발생했지만 가장 최근 데이터를 읽는 업무의 경우 암호화로 인한 성능저하가 다른 업무 유형에 비교해

적었다.

워크로드 F는 Read-Modify-Write가 발생하는 경우로서 저장된 데이터를 읽고 변경 후 다시 저장하는 조건으로 부하를 발생시킨 결과이다.

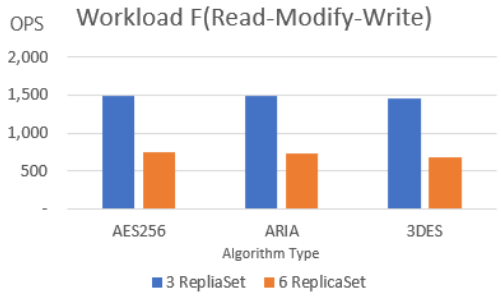

Fig. 20. Read Modify Write Measurement Results

컬렉션의 도큐먼트를 Read-Modify-Write와 같은 업무 형태에 대해 테스트 했을 때 3DES 알고리즘은 데이터 증가에 따른 성능 감소가 가장 많이 발생했다.

특히, 몽고DB의 리플리카 셋을 3 -〉 6개로 증가시켰을 때 50% 정도의 OPS 성능저하가 발생했다. 데이터를 읽은 후 변경하여 저장하는 업무의 경우 암호 알고리즘에 따른 성능 변화 정도에 심했다.

VII. 결 론

몽고DB를 이용하여 빅데이터 플랫폼 환경에서 블록기반 암호 알고리즘의 성능 변화 추이를 분석하였다. 데이터 건수를 증가시키면서 알고리즘별로 성능 변화 추이를 분석한 결과 AES와 ARIA는 데이터양 증가에 대해 비례적으로 성능이 떨어졌지만 두 알고리즘간의 차이는 미미하였다. AES와 ARIA는 SPN(Substitution Permutation Network) 구조로 설계된 알고리즘으로 전치와 치환을 이용함에 따라 암·복호화 시 성능이 비슷하게 나온 것으로 생각된다. 반면, 3DES는 3개의 키를 사용하여 DES 알고리즘을 3회 반복함에 따라 데이터를 메모리로 로딩시 암·복호화 과정을 반복적으로 수행함에 따른 성능저하로 생각된다. 특히 3DES는 데이터 증가에 따라 성능저하가 가중되는 것은 이러한 알고리즘의 특성이 반영된 결과라고 할 수 있겠다.

IT시스템의 업무 중 업데이트가 빈번하게 발생하거나 최근의 데이터에 대한 읽기가 많은 경우 데이터 증가에 따른 성능저하가 현저히 떨어졌다.

또한, 리클리카 셋을 추가로 확장하여 워크로드별로 암호화 성능을 측정한 결과 3-ReplicaSet에 비해 47~62%까지 성능이 저하됨을 확인하였다. 이는 몽고DB 환경으로 빅데이터 플랫폼을 구축하는 경우 데이터 증가 시 수평적 확장(Scale Out) 방식보다는 수직적 확장(Scale Up) 방식으로 서버를 증설하는 것이 최적의 성능을 도출할 수 있다고 하겠다.

본 연구를 통해 보안성 강화를 위해 데이터를 암호화하는 경우 AES와 ARIA와의 성능 차이가 미미함을 확인하였다.

향후 몽고DB를 이용하여 빅데이터 플랫폼 구축 시 암호화를 위한 공공기관과 기업의 보안담당자에게 선택의 폭을 넓힘으로써 국내 개발 알고리즘인 ARIA의 활성화에 이바지할 것으로 생각한다.

References

[1] Byung-Yoon Sung, Ki-Bbeum Kim, and Kyung-Wook Shin, "A Cryptographic Processor Supporting ARIA/AES-based GCM Authenticated Encryption", Journal of IKEEE 22(2), pp.233-241, Jun. 2018

[2] NIS(National Intelligence Service), "Verification Object of Encryption Algorithm",http://nis.go.kr:4016/AF/1_7_3_1.do, 2019

[3] Korea Ministry of Government Legislation Center, https://www.lawmaking.go.kr/lmSts/nsmLmSts/out/2000002/detailRP, 2019

[4] Korea Legislation Research Institute, "A Study on Improvement of Personal Data Protection Law Related to Big Data", Oct. 2017

[5] Wikipedia, "MongoDB", https://ko.wikipedia.org/wiki/MongoDB, 2019

[6] M.W. Grim and A.T. Wiersma,

"Security and Performance Analysis of Encrypted NoSQL Databases", Security of Systems and Networks, University of Amsterdam, pp. 10-15, Feb. 2017

〔7〕 Min-Gyue, Jung, "A Study on Data Input and Output Performance Improvement of MongoDB and PostgreSQL in the BigData Environment" The Master's degree, The graduate of Soongsil University, pp. 33-43, Jun. 2014

〔8〕 Eun-Ki, Kim, "Research on Utilizing Nosql by Comparison of Processing Large Scale Data in MongoDB and MySQL" The Master's degree, The graduate of Soongsil University, pp. 27-28, pp 34-47, Jun. 2016

〔9〕 Sung-Soo, Jung, "I/O Workload Characteristic Analysis on NoSQL Databases" The Master's degree, The graduate of Hanyang University, pp. 27-35, Feb. 2015

〔10〕 Young-ho, Song, "Design and Implementation of HDFS Data Encryption Scheme using ARIA Algorithms on Hadoop" The Master's degree, The graduate of Chonbuk National University, pp. 19-24, Feb. 2016

〔11〕 Seon-young, Park and Youngseok Lee, "A Performance analysis of Encryption in HDFS", The Korean Institute of Information Scientists and Engineers, Journal of KISS: Databases 41(1), pp. 21-27, Feb. 2014

〔12〕 NIST, "Announcing the Advanced Encryption Standard(AES)," FIPS PUB 197, 2001

〔13〕 NIST, "Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher", SP 800-67 Rev. 2, 2017

〔14〕 IETF, "A Description of the ARIA Encryption Algorithm", RFC 5794, 2010

〔15〕 DB-ENGINES, "NoSQL Database Usage Rankings", http://db-engines.com/en/ranking, 2019

〔16〕 MongoDB, "MongoDB Architecture", http://mongodb.com/mongodb-architecture, 2019

〔17〕 MongoDB, "Security-TLS/SSL(TransportEncryption)", https://docs.mongodb.com/manual/core/ security-transport-encryption/, 2019

〔18〕 MongoDB, "Secruity-Encryption at Rest", https://docs.mongodb.com/manual/core/security-encryption - at-rest, 2019

〔19〕 NIA(National Information Society Agency), Open Data Portal Stipulation, https://www.data.go.kr/dataset/fileDownload.do?atchFileId=FILE_000000001524257&fileDetailSn=1, 2019

〔20〕 OASIS, Key Management Interoperability Protocol(KMIP) TC, https://www.oasis- open.org/ committees/tc_home.php?wg_abbrev=kmip, 2019

〔21〕 MongoDB, "Security-Configuration Encryption", https://docs.mongodb.com/manual/tutorial/configure-encryption, 2019

〔22〕 Github, "Yahoo! Cloud Serving Benchmark", https://github.com/brianfrankcooper/YCSB, 2019

〔23〕 Brian F.Cooper, Adam Silberstein, Erwin Tam, Raghu Ramarkrishnan, and Rushsell Sears, Yahoo! Research Santa Clara, "Benchmarking Cloud Serving Systems with YCSB", pp. 5-11, 2010.
〔24〕 Wikipedia, "Zipf's law", https://ko.wikipedia.org/wiki/Zipfian, 2019
〔25〕 Ki-Sung, Kim, "Performance Comparison of PostgreSQL and MongoDB using YCSB, Journal of KIISE 43(12) pp. 1385-1395, Dec. 2016

〈저자소개〉

이 선 주 (Sunju Lee) 정회원
1998년 2월: 전남대학교 무역학과 졸업
2017년 9월~현재: 고려대학교 컴퓨터정보통신대학원 소프트웨어보안학과 석사과정
〈관심분야〉 개인정보보호, 네트워크 보안, 빅데이터 보안

허 준 범 (Junbeom Hur) 종신회원
2001년 2월: 고려대학교 컴퓨터공학 졸업
2005년 8월: 한국과학기술원 전산학 석사
2009년 8월: 한국과학기술원 전산학 박사
2009년 9월~2011년 8월: University of Illinois at Urbana-Champaign 박사후 연구원
2011년 9월~2015년 2월: 중앙대학교 컴퓨터공학부 조교수
2015년 3월~2016년 8월: 고려대학교 컴퓨터학과 조교수
2016년 9월~현재: 고려대학교 컴퓨터학과 부교수
〈관심분야〉 응용 암호, 네트워크 보안, 클라우드 보안, 시스템 취약점