데이터베이스 보안 강화를 위한 재암호화 성능 분석

엄시우*, 김현준*, 송민호*, 서화정**

*한성대학교 (대학원생)
**한성대학교 (교수)

Re-encryption Performance Analysis to Strengthen Database Security

Si-Woo Eum*, Hyun-Jun Kim*, Min-Ho Song*, Hwa-Jeong Seo**

*Hansung University(Graduate student)
**Hansung University(Professor)

요 약

본 연구는 양자 컴퓨터의 잠재적 위협에 대응하여 데이터베이스 보안을 강화하기 위한 재암호화 방법을 분석하고 최적화 방안을 탐색하였다. AES 알고리즘을 사용한 실험을 통해 키 길이 증가가 성능 저하에 미치는 영향은 크지 않다는 것을 확인하였으나, 재암호화 과정의 시간 소모와 데이터 노출 위험이 있는 것에 대해서 확인하였다. 이를 해결하기 위해, TEE와 KMS과 같은 기술을 활용하는 보안 방안과 GPU 가속을 사용한 처리 속도 향상 방안을 제안하였다.

I. 서론

세계는 디지털화의 시대를 살고 있다. 정보와 데이터는 새로운 '디지털 원유'로 간주되며, 이를 기반으로 하는 산업과 기술이 폭발적으로 성장하고 있다. 그 중심에는 데이터의 안전한 저장, 전송, 액세스가 요구되는데, 이를 위해 암호화 기술은 필수적으로 사용되고 있다.

하지만 양자 컴퓨터[1]의 발전과 함께, 그 동안 사용되던 전통적인 암호화 알고리즘들의 보안 강도가 상대적으로 약해진다는 지적이 이어지고 있다. 양자 컴퓨터는 특정 암호 알고리즘을 깨뜨리는 데에 기존 컴퓨터보다 월등히 빠른 성능을 보이기 때문에, 이를 대응하기 위한 새로운 암호화 방식이나 기존 방식의 개선이 요구된다[2].

이러한 암호화 방식의 변화는 데이터를 중심으로 하는 다양한 IT 시스템, 특히 데이터베이스의 성능과 자원 사용에 큰 영향을 미칠 수 있다. 예를 들어, 키 길이를 늘려서 보안 강도를 높이는 경우, 그에 따른 연산과 저장 공간의 증가로 인한 성능 저하가 우려된다.

본 논문에서는 이러한 변화의 중심에 있는 데이터베이스의 성능 및 자원 사용에 대해 분석하고, 이를 어떻게 대응해야 하는지에 대한 방안을 탐구하고자 한다.

II. 성능 분석 방법

데이터베이스 관리 시스템(DBMS)는 복잡한 데이터 구조와 연산을 효과적으로 관리하기 위한 시스템으로, 사용자의 편의를 중심으로 다양한 기능을 제공한다[3]. DBMS는 대규모의 데이터를 저장, 검색, 갱신 및 관리할 수 있도록 설계되었으며, 다중 사용자 환경에서도 데이터의 일관성과 무결성을 유지한다. 또한, DBMS는 데이터의 안전성을 보장하기 위한 다양한 보안 기능을 갖추고 있다. 대부분의 DBMS는 내장된 암호화 함수를 제공하여, 사용자 데이터나 중요 정보를 보호한다.

데이터베이스 성능 분석을 위해, 주로 사용되는 SELECT문과 INSERT문을 중심으로 암호화된 데이터의 입력 및 조회 성능을 측정한다[4]. 안전하게 데이터베이스에 정보를 저장하기 위해서는 저장하기 전에, 암호화 과정이 필수적이다. 이 과정에서는 개발자의 선택에 따라 사용자 측에서 암호화를 진행하거나 DBMS에서 직접 암호화하는 두 가지 방식이 있다.

DBMS 중심의 암호화 방식에서는, DBMS 내부의 암호화 함수를 통해 데이터가 암호화되어 저장된다. 또한 데이터를 조회할 때도 DBMS에서 해당 데이터를 복호화하여 사용자에게 제공한다.

반면, 사용자 중심의 암호화 방식에서는, 사용자가 데이터를 DBMS에 전달하기 전에 암호화를 하고 전달한다. 데이터 조회 시에도 DBMS는 암호화된 데이터를 그대로 사용자에게 전달하고, 복호화는 사용자에서 진행된다.

또한 두가지 경우에서는 키 관리의 측면에서도 차이가 있다. DBMS 중심의 암호화에서는 DBMS가 각 사용자의 키를 관리하며, 데이터베이스에 해당 키가 저장되어 사용된다. 반면, 사용자 중심의 암호화에서는 개발자의 설계에 따라서 데이터베이스에 저장되어 사용될 수도 있지만 사용자가 직접 키를 관리할 수도 있다. 본 논문에서는 사용자가 직접 키를 관리하는 경우로 성능 분석을 진행하였다. 이러할 경우 데이터베이스에 별도의 키 저장이 필요하지 않으며 데이터베이스에서 키를 가져와 복호화를 진행하지 않는다.

2.1 성능 분석을 위한 테스트 테이블 구성

Field	Type	Purpose
userid	int	index
cipher_1	varchar	< 16-byte
cipher_2	varchar	16-byte
cipher_3	varchar	16~512-byte

Table 1. 암호문을 저장하기 위한 테이블 구성

Field	Type	Purpose
userid	int	index
user_key	varchar	used key
user_iv	varchar	used iv

Table 2. 사용된 키를 저장하기 위한 테이블 구성

<표 1>은 데이터베이스에 암호화된 데이터를 저장하기 위한 테스트 테이블이다. userid는 사용자를 구분하기 위한 index로 사용되는 속성값이다. cipher_1은 16바이트 미만의 데이터를 암호화하여 저장하는 속성이다, cihper_2는 16바이트로 고정된 데이터를 암호화하여 저장하는 속성이다. 마지막으로 cipher_3은 16~512 바이트의 랜덤한 길이의 데이터를 암호화하여 저장하는 속성이다. 데이터베이스에는 다양한 종류와 길이의 데이터가 저장될 수 있기 때문에, 여러 가지 길이의 데이터를 사용하여 성능 분석을 진행하였다.

<표 2>는 암호화에 사용된 키와 IV값을 저장하는 테이블이다. <표 1>과 <표 2>에서 같은 index일 경우 <표 1>에 저장된 암호화된 데이터는 <표 2>의 저장된 key와 iv값으로 암호화되었으며, 이 값으로 복호화를 진행한다.

2.2 데이터 입력 성능 분석 시나리오

DBMS를 활용할 경우 DBMS에서 제공하는 암호화 함수만을 사용하여 구현을 진행한다. 암호화에 사용할 데이터, 키 및 IV는 RANDOM_BYTE() 함수를 사용하여 랜덤한 값을 생성하여 사용되었다. DBMS에서 암호화를 하여 데이터베이스에 저장할 경우의 쿼리는 아래와 같다.

```
INSERT INTO
테이블명(cipher_1, cipher_2, cipher_3)
VALUES(
HEX(AES_ENCRYPT(@text_1, @key, @iv)),
HEX(AES_ENCRYPT(@text_2, @key, @iv)),
HEX(AES_ENCRYPT(@text_3, @key, @iv))
);
```

위의 쿼리문이 반복 동작하는 시간을 측정하여 성능을 분석한다. @변수명은 DBMS에서 사용되는 세션 변수로 값을 임시로 저장하여 사

용할 수 있다. text, key, iv는 INSERT문이 동작하기 전에 랜덤한 값을 생성하여 암호화에 사용된다.

반면에 사용자가 암호화를 진행할 경우 HEX(AES_ENCRYPT()) 대신에 암호화된 데이터를 그대로 전달받아 데이터베이스에 저장하게 된다.

2.3 데이터 조회 성능 분석 시나리오

데이터 조회는 index를 순서대로 조회하였을 때와 랜덤한 index를 조회하였을 때 두 개로 나누어 성능을 분석한다.

데이터 입력과 마찬가지로 DBMS의 경우 DBMS에서 제공하는 AES_DECRYPT() 복호화 함수를 사용하여 데이터를 조회한다. 데이터 조회 쿼리문은 아래와 같다.

```
SELECT
AES_DECRYPT(UNHEX(@cipher_1), @key,
@iv);
```

데이터 입력과 동일하게 위의 쿼리문을 반복 동장하고 시간을 측정하여 성능을 분석한다. 해당 쿼리가 동작하기 이전에 Key와 iv를 저장하고 있는 테이블에서 index에 해당하는 값을 불러와 @key, @iv 세션 변수에 저장하는 과정이 선행된다.

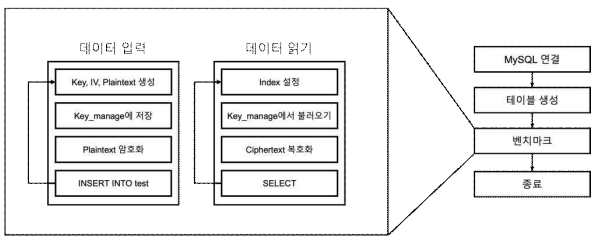

Fig. 1. 전체적인 성능 측정 시나리오 구성

데이터 입력과 조회의 전체적인 시나리오는 (그림 1)과 같다.

III. 성능 분석 결과

사용된 DBMS는 MySQL을 사용하였으며, 구현은 Python을 통해 구현하여 성능 측정을 진행하였다. 사용자가 암호화를 할 때 사용된 외부 암호화 함수는 Python으로 사용 가능한 pycrptodome 모듈을 사용하였다.

암호화를 위해 사용된 암호 알고리즘은 가장 널리 사용되는 AES 알고리즘을 사용하였으며, 128-bit 길이의 키를 사용할 때와 256-bit 키를 사용할 때의 성능을 분석하였다. 이때 블록 암호 운용 모드는 CBC 모드를 사용하였다[5].

구현 환경은 8.1.0ver MySQL을 사용하였으며, Apple M1칩이 장착된 MacBook Pro 13-inch 2020 상에서 성능을 측정하였다.

성능 측정은 10,000번 반복 동작하였을 때의 시간을 측정하였으며, 결과는 아래 <표 2>와 같다.

	AES-128	AES-256
DBMS-입력	5.70	5.48
DBMS-조회	14.60	14.58
DBMS-조회*	13.94	13.94
사용자-입력	6.02	6.04
사용자-조회	1.27	1.34
사용자-조회*	1.28	1.35

Table 4. 암호 알고리즘에 따른 성능 측정 결과 (*: 랜덤한 Index, 단위 : 초)

<표 2>의 결과에 따르면 보안 강도를 높이기 위해서 키 길이를 늘렸을 경우, 키 길이가 늘어남으로 인해 암복호화를 위한 라운드가 증가함으로 인한 연산량이 증가함에도 불구하고 성능의 차이는 거의 동일한 것을 확인할 수 있다.

하지만 30,000개의 데이터에 대한 테스트 결과라는 점에서 데이터 입력과 조회에 많은 시간이 소요되는 것을 추가적으로 확인할 수 있다. 만약 AES-128로 암호화된 데이터를 보안 강도가 높은 AES-256으로 다시 암호화하여 저장하게 될 경우, AES-128의 조회 성능 결과와 AES-256 입력 결과를 더하여 보았을 때 DBMS의 경우 약 20초, 사용자의 경우 약 7초의 시간이 소요되는 것을 알 수 있다. 이는 실제 환경에서 사용되는 데이터 수에 비하면

30,000개의 데이터는 매우 작은 수의 데이터이기 때문에 더 많은 시간이 필요하다는 것을 예상할 수 있다.

또한 재암호화를 위해서 암호화된 데이터를 복호화하여야 하기 때문에 원본 데이터가 노출될 수 있는 위험성이 존재한다.

결론적으로 키 길이를 변경함으로 인한 성능 차이는 미미하지만 재암호화를 위해 많은 시간이 소요되며, 원본 데이터가 노출될 수도 있는 문제가 있다는 것을 알 수 있다.

따라서 본 논문에서는 4장에서 안전하게 재암호화를 하는 방법과 좀 더 빠른 재암호화 방법에 대해 제안한다.

IV. 재암호화 과정 문제 보완

4.1 안전한 재암호화 방법 제안

데이터베이스에 저장된 데이터를 재암호화를 하기 위해서는 복호화 과정이 필수적이다. 이로 인해 재암호화 과정에서 원본의 데이터가 노출되는 문제가 있기 때문에 안전하게 재암호화를 하기 위한 방법을 두가지 제안한다.

첫 번째는 TEE(Trusted Execution Environment)를 사용하는 방법이다. TEE는 보호 영역을 설정하여 CPU가 메모리 일부를 암호화하고, 보호 영역에 있는 데이터를 보호 영역 내에서 실행되는 프로그램 이외에는 접근할 수 없도록 하는 기술이다[6]. 이 기술을 활용하여 데이터가 복호화 되더라도 외부의 접근으로부터 안전하기 때문에 안전하게 재암호화가 가능하다.

두 번째는 키 버전을 관리하는 방법이다. KMS(Key Management System)은 키 생성, 저장, 분배, 순환과 같은 기능을 제공하는 시스템이다. KMS에서는 이러한 기능을 제공하기 위해서 키 버전을 관리하고 있으며, 안전한 키 관리를 위해서 주기적으로 키 변경을 권장하고 있다. 이때 키가 변경될 때, 키에 대한 버전을 관리한다. 이 버전을 활용하여 어떠한 데이터가 어떤 키로 암호화 되었는지 알 수 있다. 이러한 방법을 활용하여, 한번에 재암호화를 하는 것이 아니라 사용되는 데이터를 다시 암호화 할 때 새로운 키로 암호화 하는 방법으로 순차적으로 조금씩 재암호화를 하는 방법을 제안한다.

4.2 빠른 재암호화를 위한 방법 제안

3장의 결과를 통해서 알 수 있듯, 적은 데이터를 재암호화를 할 때에도 많은 시간이 소요되는 것을 확인하였다. 실제로 저장되어 있는 데이터는 실험에서 사용된 데이터 보다 더 큰 데이터가 사용되고 있으며, 이로 인해 더 오랜 시간이 소요되는 것을 알 수 있다. 또한 주기적으로 키를 변경해주어야 하기 때문에 매번 많은 시간이 소요되는 문제도 해결되어야 한다.

이러한 문제를 해결하기 위해서 GPU를 활용하여 재암호화를 하는 것에 대한 방법을 제안한다. GPU는 CPU보다 더 많은 연산을 병렬로 실행하는 것이 가능하기 때문에 많은 데이터에 대해서 CPU보다 빠른 속도로 연산하는 것이 가능하다.

데이터베이스에서 재암호화를 할 경우 모든 데이터들은 동일한 연산을 하기 때문에 병렬로 구현하는 것에서 많은 이점을 얻을 수 있다. 최근에는 BrylyDB, SQreamDB, HEAVY.AI와 같은 GPU를 활용하는 데이터베이스 플랫폼도 개발되고 있으며, 실제로 데이터베이스에서 GPU를 사용함으로써 36배 빠른 속도를 보여주는 연구 결과도 확인할 수 있다[7].

V. 결론

본 연구에서는 양자 컴퓨터의 등장이 현존하는 암호화 기술에 미칠 영향과 데이터베이스 보안을 강화하기 위한 재암호화의 중요성을 탐색하였다. AES 암호화 알고리즘을 사용하였을 때, 키 길이의 증가가 성능에 미미한 영향을 미치는 것을 확인하였으나, 재암호화 과정이 상당한 시간을 요구하고 원본 데이터의 노출 가능성을 확인했다. 이에 대응하여, TEE를 활용한 보안 강화와 KMS에 의한 키 관리 시스템을 제안함으로써, 데이터베이스의 안전한 재암호화를 위한 기술적 해결책을 모색하였다. 또한, GPU를 활용한 재암호화 방법을 통해 처리 속도를 향상시키는 방안을 제시하였다. 4차 산업 혁명 시대를 대비하여, 대규모 데이터베이스 시스템에 안전하고 효율적인 재암호화 프로세스의 적

용은 필수적이며, 이에 따른 지속적인 연구와 기술 발전이 요구된다.

VI. Acknowledgment

This work was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIT) (No.2018-0-00264, Research on Blockchain Security Technology for IoT Services, 50%) and this work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.2022-0-00627, Development of Lightweight BIoT technology for Highly Constrained Devices, 50%).

[참고문헌]

[1] Steane, Andrew. "Quantum computing." Reports on Progress in Physics 61.2 (1998): 117.

[2] Mavroeidis, Vasileios, et al. "The impact of quantum computing on present cryptography." arXiv preprint arXiv:1804.00200 (2018).

[3] David Maier, Jacob Stein, Allen Otis, and Alan Purdy. 1986. Development of an object-oriented DBMS. SIGPLAN Not. 21, 11 (Nov. 1986), 472-482. https://doi.org/10.1145/960112.28746

[4] Date, Chris J. A Guide to the SQL Standard. Addison-Wesley Longman Publishing Co., Inc., 1989.

[5] Daemen, Joan, and Vincent Rijmen. "AES proposal: Rijndael." (1999).

[6] Sabt, Mohamed, Mohammed Achemlal, and Abdelmadjid Bouabdallah. "Trusted execution environment: what it is, and what it is not." 2015 IEEE Trustcom/BigDataSE/Ispa. Vol. 1. IEEE, 2015.

[7] Bakkum, Peter, and Kevin Skadron. "Accelerating SQL database operations on a GPU with CUDA." Proceedings of the 3rd workshop on general-purpose computation on graphics processing units. 2010.