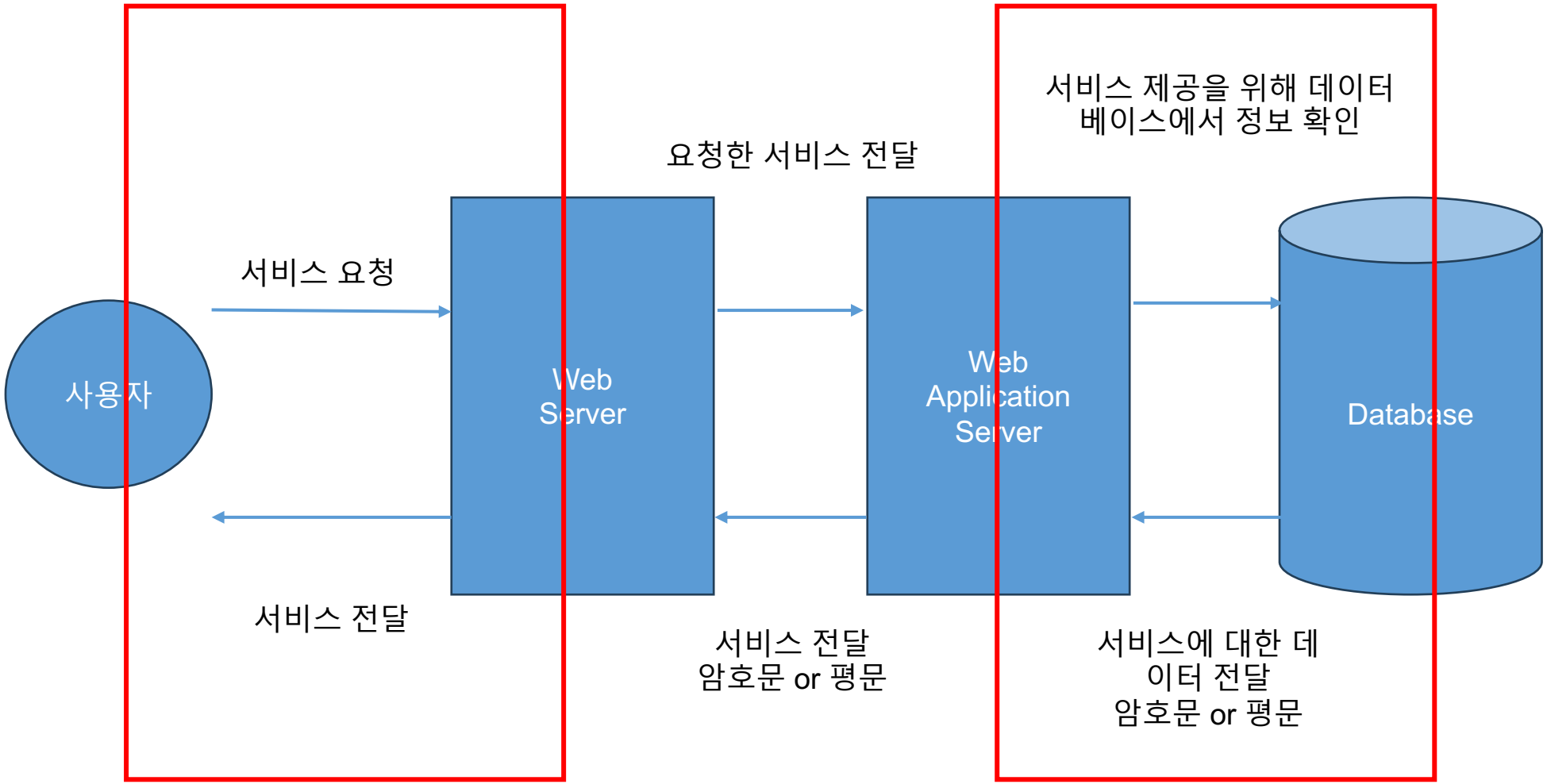
DB 암호화 Bench test 구현

https://youtu.be/ZNcGzhaiGWY

1. Bench Test 시나리오

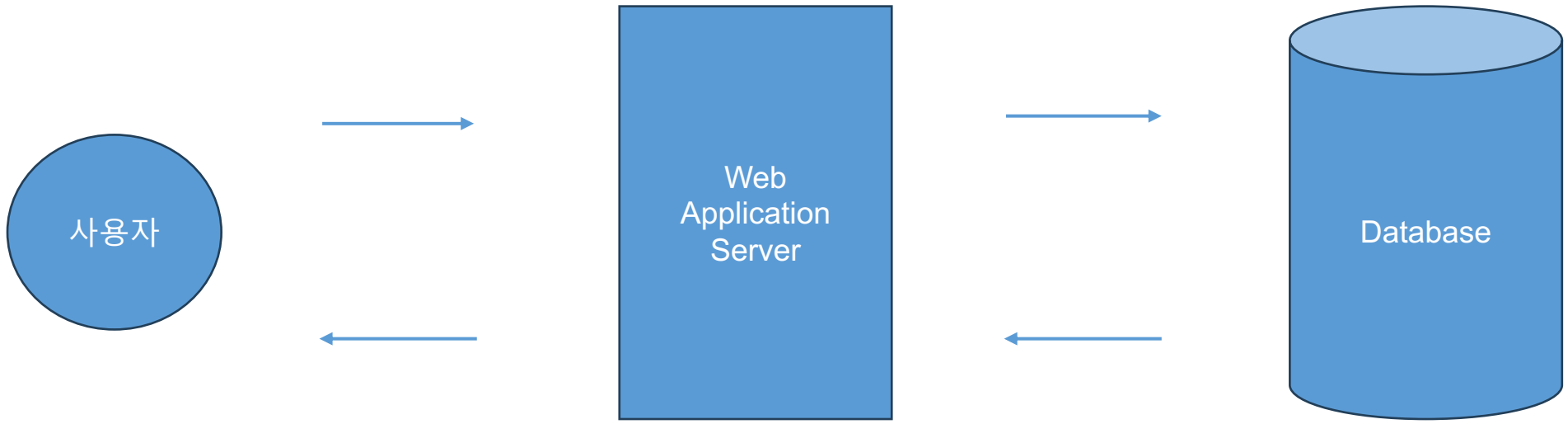


1차 암호화
사용자와 서버간 통신을 위한 암호화

2차 암호화
서버에서 데이터베이스에 저장하기 위한 암호화

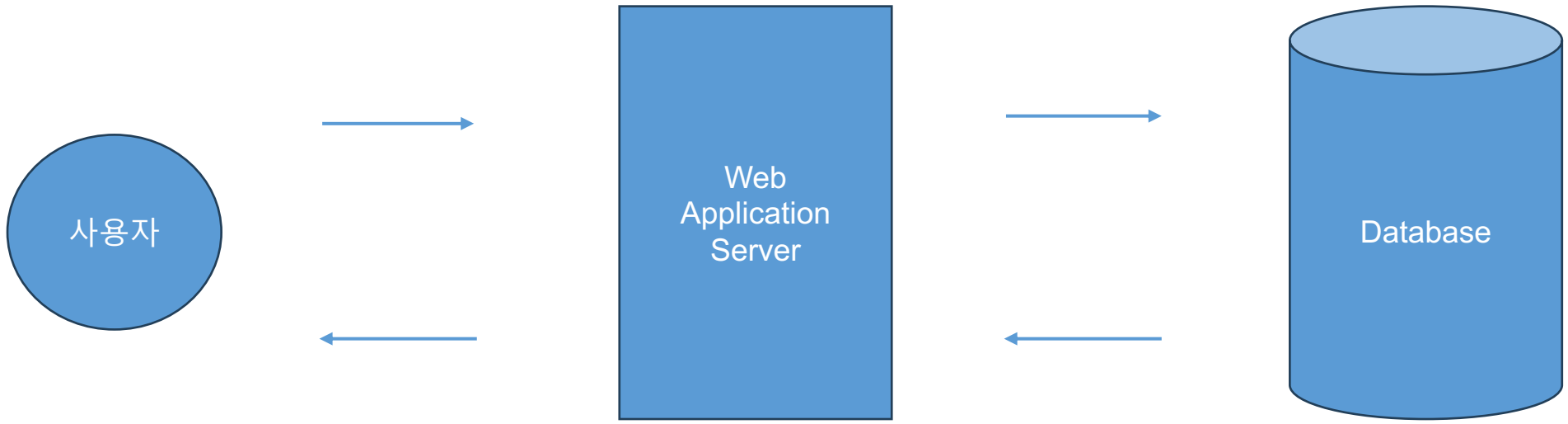
1. Bench Test 시나리오

- MySQL 내장함수 사용
  - 서버에서 암호화를 하여 데이터베이스에 저장하는 방법
  - MySQL DBMS에서는 암호화 함수를 지원함
    - AES_ENCRYPT(), AES_DECRYPT()를 활용함

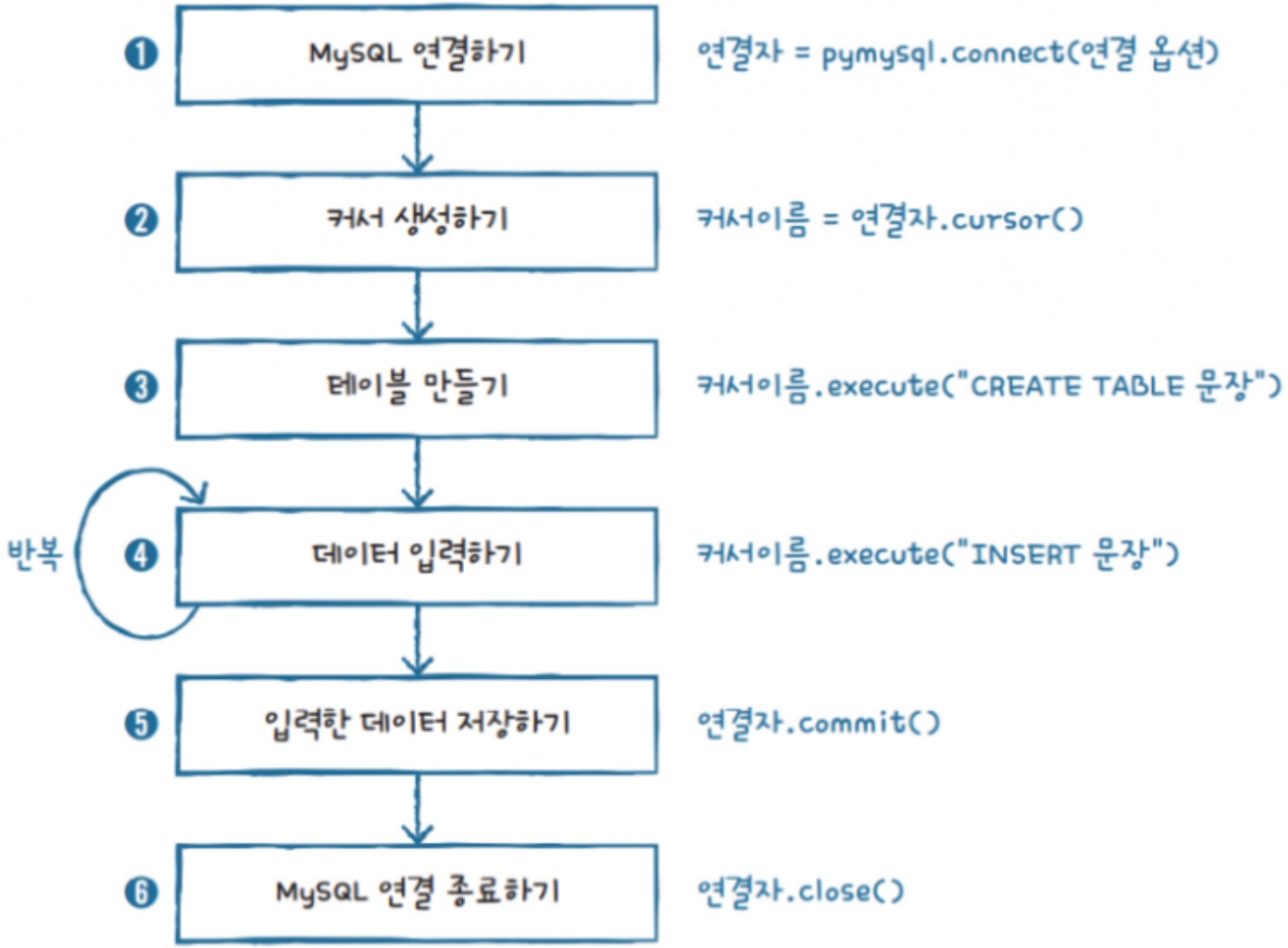


1. Bench Test 시나리오

- Python 라이브러리를 사용
  - 사용자가 암호화를 하여 서버로 전송하고 서버에서는 데이터베이스에 저장만 하는 방법
  - Python의 Crypto 라이브러리를 사용



1. Bench Test 시나리오



2. Table 구성

```
table_init = ("CREATE TABLE test("
              "userid int auto_increment not null primary key,"
              "ciphertext_1 varchar(256),"
              "ciphertext_2 varchar(512),"
              "ciphertext_3 varchar(3072)"
              ")CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;")

table_key = ("CREATE TABLE key_manage("
             "userid int auto_increment not null primary key,"
             "temp_text_1 varchar(256),"
             "temp_text_2 varchar(512),"
             "temp_text_3 varchar(3072),"
             "user_key varchar(64),"
             "user_iv varchar(64)"
             ")CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;")
```

Test 테이블 : 암호화된 데이터를 저장하는 테이블

userid	ciphertext_1	ciphertext_2	ciphertext_3
int	Varchar(256)	varchar(512)	Varchar(3072)
Index로 사용	16바이트 미만의 랜덤한 작은 데이터	16바이트의 고정 데이터	16~512바이트의 랜덤한 데이터

Key_manage 테이블 : 암호화에 사용되는 Key, iv를 저장하는 테이블

userid	Temp_text_1	Temp_text_2	Temp_text_3	User_key	User_iv
int	Varchar(256)	varchar(512)	Varchar(3072)	Varchar(64)	Varchar(64)
Index로 사용	암호화 이전의 평문	암호화 이전의 평문	암호화 이전의 평문	사용된 key	사용된 iv

```
mysql> DESC TEST;
+-------------+--------------+------+-----+---------+----------------+
| Field       | Type         | Null | Key | Default | Extra          |
+-------------+--------------+------+-----+---------+----------------+
| userid      | int          | NO   | PRI | NULL    | auto_increment |
| ciphertext_1 | varchar(256) | YES  |     | NULL    |                |
| ciphertext_2 | varchar(512) | YES  |     | NULL    |                |
| ciphertext_3 | varchar(3072) | YES  |     | NULL    |                |
+-------------+--------------+------+-----+---------+----------------+
```

```
mysql> desc key_manage;
+-------------+--------------+------+-----+---------+----------------+
| Field       | Type         | Null | Key | Default | Extra          |
+-------------+--------------+------+-----+---------+----------------+
| userid      | int          | NO   | PRI | NULL    | auto_increment |
| temp_text_1 | varchar(256) | YES  |     | NULL    |                |
| temp_text_2 | varchar(512) | YES  |     | NULL    |                |
| temp_text_3 | varchar(3072) | YES  |     | NULL    |                |
| user_key    | varchar(64)  | YES  |     | NULL    |                |
| user_iv     | varchar(64)  | YES  |     | NULL    |                |
+-------------+--------------+------+-----+---------+----------------+
```

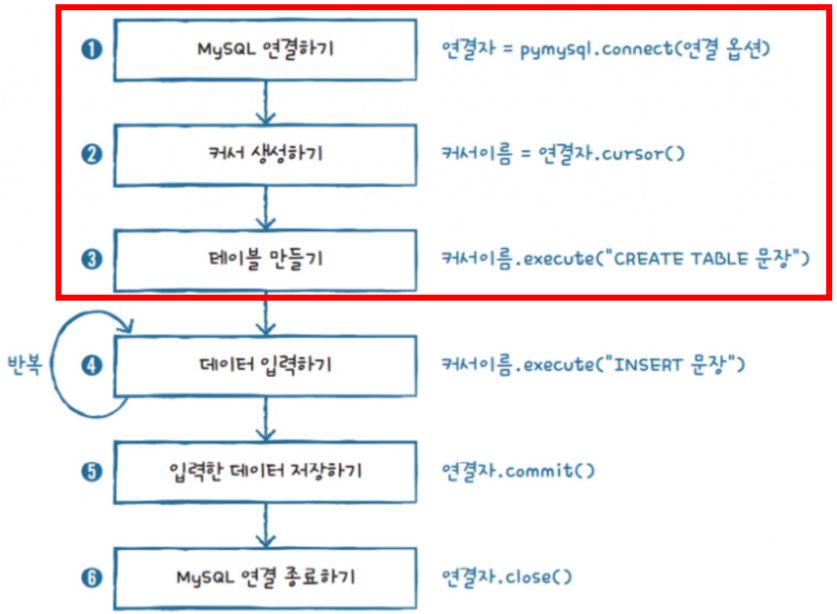
2. Table 구성

userid	ciphertext_1	ciphertext_2	ciphertext_3
1	576588ef7e51ccc34ff5b6bfc30b1969	4946080c7e79e8dfe43355b7480658096d6c560812b7120f4b507918e4189a12	d67d197eeee7a7dccebа6e7b5d91240d9e90b2e3345e98a798e5da6ea340e475a36d270d5bb4df51429356da94199b9488bb72d066b93c2da6f3ed34f8909d25
2	463ad09996ee225b0a85f3734782178d	4a2265b60b6ed01fc0885770523fcd0e4e79a9683d981c7bff0f573b0740854f	c0eb1b423c8b5d3f1d76984c6c1bd390c85be823742ed2262cf3476c7459ef13c5d0b13155439c46101f79d8029bbc60ebcb2cc9f86ff504224d3953fcd377a66cc311fb77b5b6bc3b90c36ed25f85b18d98cd698d0c2c751542e83dc36112ffc9ef316fbcb6d3593d2fd7706421d069ba6d745d0975b5dec3cc0b9eef92cb2226611a3e7a163f1c0cf0e7dfae3050af38cc7c53fa02341b216683d7106524e1215a4c991d522ee1a7aba38ffab9e29557a8a500e5922131b370790d0dc8d855c9b3a55a09dae307d136a064f29bbdc6
3	d95b3582f1e9e7bff78257a735b64907	e021d9fa5b7ab383e3f7dd404b9af146455de6ee7ec1a18a899b2b08fede801f	346c136dbd1a8ac0b7e1b2fb76ded4eec0385a71f6c36ec56912c21d2a00eb49b2b3c899fd9d3980f5c522395fcf377d352a02c3711b66d979889464074f098b2bdcafbd08ac56ceeb9ad5c3e067080a8d30011d14d1541589b7881f64e79e783cde17c4ccc7aad12fea6cac7a98b95e52bf45a4fcaa2676de4ebc5b0b1aa2ebd6475e580fbc47c6afafb3e67df6e9e2a0401beeb5c1bc6550b1bf6fce4cff5fe0e26a6ff0308fe057cae274f1a8110db1a9e29fe719f540e95213029aca9500074a82bc998fd1f655f495fdcb2bb0f910cf576bb955fd56c73068449dd3755e39ab727fa401de43e69e81729fbc1e47a0a8dfceb574ebe3102f942b8e33d331a5c97f960619f4875f6e4421286210670d41f4c70db381e0609ee1fcacf4777a5e1a4c1fde090bce4f74e9aec09ec88acb1209ca9db24bda07367e6b65b3a9a63bfa03d68e75d61f8ff9c187dd62c0bb490bd8adf75c5be5557faf632ac2d683b4f4354a9fe0aa6e3293dbec65904d95557d1cb8a1ae637b48981cba21875036236cbf935af0d5d96de6c63d2dd7d08152c8d022fcbb39396c78ab1e5370e17081f21c4885ab26852e0c1643369dd6f9d0c0f02c8c7f3286e89ee375bcb024ccadc39c0abe58b556a2cf4236f47ac59b3d270920a8a059e408fd95e706a3de4ec01e696a45245a137855bfea9e7361a
4	1571242d3dfff43dc3557cf2b77b1ff0	91d7d06deadee0382745df92de93340ffb47e12695a41e7039fbf8c9bbc5cc66	ecdd5d3784e8f06903be2e3a042fb81459ef1237b583d3f5ba65ff307a2a55cbbe7e7d6e9df4f4d48629450c75d63ace4ec25d61240e6b99e8c2598dad2d07230f7888678acfcc6c7524924b8b65b21f9821be151baab5f3076173204ce7290afb41e36413a4dfe8be4a3a347eb510bc92006dfd5bf9749dce08cdfee503dae3fe879453e2f74eed9545f687cd96c5eee824a115969669db980912b87d43eed49ae2d8aae52fd133a0b2aa784d283eb1a05e8d1c2bdd99beb9fbe38283df765e42378b3ec1ff451a3ad98a207b82e8e2
5	f1ea2906d7af0d0bc05d553a272a768f	b5f2ec634bbd2539d0219a2d326f38357783c8407598778b9bffe32c16e916d3	70ec45b7f4d40fccb26b2ba92ab78cda401a48a944ad50a3a5142123adf5cd0e50c184c6c2a2a54a8f416303918eaca75b73772e78c67328ca1ffae135899678699d4a0d4a540cffc2ccaa5cc3b1a7028df6face360e7f3d1be6c44758d917e8f3931cdb8f6e3fbef40980b081ca471b5bd79c22bd3da4bcf6bad66d3264d83047bf9b8b38b82b57ce53ce87e19f019d58a6f7293e1951a51ea6b45cab7751601eb79032ae379b53d174ee8ebf230c6167a3dcfd7d159ec4219e5f70d69be27d306ff63607012184b3d4eac98ed23beb705d9132c7cb1a2fb428f0d8a556c12d572c6c6fa649375ad06d14a6f24933ec3d28f48b63982af6796714ab7e59b26ba13c5510944242b4c01e9e5d1f7738b996cda7461b5a8c0e2194ad9e07bfc4ea58bcde008e47db696d4e5690a9c829055695aa37609d5931c5357540cf3114050c32da48129b1baaa98314c74da373cbc38a7b0ce682d14ee4c37eb6d1e6488b97a007df705dea6cd49410802dc7c24c0a4db980ae3205ddf71e0e3c0b5fa31a943cc3d5a1c5224625e2d74548dac0bcf277e85a38254d7d337bbfc20b3e57cb69d69ea0a3a24f9bfa26f8417c9a7c132170f2360881dad061d0b7089a6ad8da8a2579d3b8ef589dbc87c56dcd8648e
6	f9dd3b3a5f0d687c380eae8b4c4d58d0	483f9a3b86cb88d664f660e716e7a776514396588e7b0d8f3bb2bf95447d3646	6624081e9ee0120b03b7344302f76a615c978bf4905d64263262f8019a3b0c02856055d3ee8a76dcbd2c12d3a2ab22a6b8780a458d61fd1fade81a9049be7e3e3371f6811297595e8e957d41bebdec03d4d8e778756508ee6f8e9518aa18869fca51efb96072b848f360007bc69915bd2b47d681af9b1b990696dd06763fe995c38c320e9e79d8eb3d52ef839d445efc925b6669f1c0ab752ad5828723d7cb3b5d1d9a50d6cba60b8f864b461c292598cf2b08ff17c626335ecccda8e2001630
7	f4b51a1a8d5050ea69f2cb499d1a7905	1fc7f4050c474149846ab75983a635b15fbe9030c4c857d876bb6d34dfe5c73b	11e15782af3de587184599c8d8b4c395de0c35b95d79cfb4813774355558c0822ff454dbda24b4e98df91e51fd4ea207ee40d9a0ef7a4e12aa86b3ddd68e7cb7dd58ae4aa2fd4c564f9d29a996e1b90fa7ab9e8b6379de34241b9a3c346fa3ead5c1da8e2c1ba18529e209efbd0f0273edd563976995fe5256110696a51d7c12a195df7c6c947c1741f14091b84893b850a1fd2ee72a8196c83ecc5278f017f4ca90ac7e2810cc1454c944e7b48a3a0ae6245d1905cd1e09279c5e3f77834d3a85200fcdfbb29824fdf8252817717ef4a06c3a33c5fa88c8ae13bb33825430d17d26e385079158f614f5ede0a304dd75c12e543d5deb3cf49717b858e14ce3047607754768 44cafe8dcabcf58a0d92f8f7b694c27288906da9e0e00680b6085a6d16771a463bfa15f0fef397727f48ea372454598f613b84e848a215de73f7f53569504abc5d7e236b481b29e1d1498b38e8ddb7aefcf514c0b4717b2e200410fcc5ae5ff11a247c4d8931af465bcc406e40853efd2c3e987f7d52845e5ac73bd5397d6a9a5c67dc6bfd874c3dc77dcaddcf7dc3ea6c9a7dae71c6f4474b3c035a3a08a0285cd3115394e6598a9d7c74d26fa8d886f2bc4899db2ec2a4ab521ae36464363dde08c3074a403c75b507ac481ff3cef2bccc13987237c7748c88b5cf4a64c752f4a79cea745b8b7f1238d6af26c085a514cbe88002d6381527f83d
8	eb59fa11592d5a2842ed5f471fef2a4e	c5e009cd1eb30e82fd059f9b72e4a7d5403a22250d947d5b8144d168002b575b	81fff9f5a49317cea44e438bb95742b5deddba9487ed4a4e5c0a847fbcd380fb632c185e95c10c14980f6f888bf387f71e0f1c8aab7b6fdeec2848019563c5241efa49641231c2376fbe6c76edfe7e0c
9	0faf654d11c3dd903a47594b5cfffcf5	9f8abe5726e1a3208d2dca52018865ae37fa1957228a177d341f1e9f33530319	d5890621dfe0fba627f01aef8b3f8d55bdf21ca417aaf46656f34cbd17df59779070e8c5e26f9d9fed596a98be0468d4681bb9e4ef3d7da4d97811a05d5d6020c8f4435f7a21e9da799eb09df4e9cb3211509aca8e743198d7145852f768ac9f43379f5e5e71a562305a1782a20bb9875239c5faba95dc1d4227e02b4fea3dc04492020ea4bc08bfd68f7377523f496ab82393f75b11272ab5227f799e104fae7b4ca155a2759bf20d7bd8057c8f62676dba3ed4c884341ae11c28b54de135fbb7ae065a2efb04733f10a83f706ed03309edbc57d6a353a2326458a89ca82de949cb1a3e31075a462962ac7f8cea74b7f707eb58437fa0d56d1ed2f96180ba1aaa0aae678a876f5d26c71a479355a513c2644443d7a2da2934a0bc3b24fecb535726f9939bdbcdaeddba699f9befd0a3c89daddff8e3eebeafcb4e2cb349983364ff48888f84d30813618cc95f1c3f9e87863fa5777316a71a62d4db30508f35f0a45984b3d92e812aafbf4bdc6417d4076bf00badbc333f0b639e04652b41b1d0a32e0967f848375e71dd29ff4c06d6
10	6c72dd5e6ebd31b63e577a54f5d9ef24	f1f04404be189330bb126e4973da94bc98dd86fe4a4bd50fa2ded9ea4d6f22e1	2fa5f20c35fff6e381c12c305e488ea4d43ed2fc22ec8a43b6dcca98cd46dc1ea6af6033c7166406c1a175eeb4396f77efa7d00073dd7fe386be8b56b7f2da9a9691ddf12884519b201c23697a93b3c9f635f87d20be4d716cd28748dae75ad2dc2bbb9cb676fe34dc1352cd3b0f00137ceb418ce59c481bd5eb0ae2881485d7d4ece04118c9ef1749f374e3d4355187aa739864bd70f9ce176ce13f994cb0ac300705de4ce3dd88938e3b930c2948fdaf1eb67410ace380241e52563f8a96b84f03d2c844f96dd19820b44cfabc40be4a2e4101c916b9cadcf34341df63dfdd8abedc3a333b72b9bf9a510087396d5063d3d5af4ccaba18933931726 2dee8305432 4cafd342be21ee0ea87f10f934f22259f642ef409a1959824264d79952203dde65c5aca3a5e6101327de9cefa6e95fd2c78d3dab77499cda769cd7edbe38dec8b87624b69dcf48935be23a27b4ab56deaad345ba1b825496a6659402c182111981c7e044374a1a118a9b32217895d60d7c2006a23bd28d26da237f76fc17318352bf000aabfa9077dde8f44c5a17aff2018d12b58ba87451d3433d19ed8506bccfaff1d92ecb01ed13d44ba1068a86415e1d4b91d8a9e5b61ddcc773845827abb97c09fa0a17215acf04d6ab9ded3ab5dd35e76cc32ab5884050c221ef17044d1ad12720ed1e29d6797f7ca43aeb

3. 구현 코드 분석

- 구현된 Bench test
  - DB에 암호화된 데이터 입력
    - 데이터베이스에 암호화된 데이터를 입력하는 시간을 측정
    - 랜덤한 text, key, iv를 설정하고 암호화된 text를 데이터베이스에 반복 입력
    - 암호화 과정과 데이터를 입력하는 쿼리만 시간 측정
  - DB에 저장된 암호화된 데이터 읽기
    - 데이터베이스에 암호화된 데이터를 가져와 복호화하여 데이터를 읽는 시간을 측정, 읽는 방법을 두개로 나누어서 구현
      - Index 순서대로 암호화된 데이터 가져와 복호화
      - 랜덤한 Index의 암호화된 데이터를 가져와 복호화
    - 암호화된 데이터와 key_manage 테이블에 저장된 key와 iv를 가져오는 과정도 시간 측정

3. 구현 코드 분석

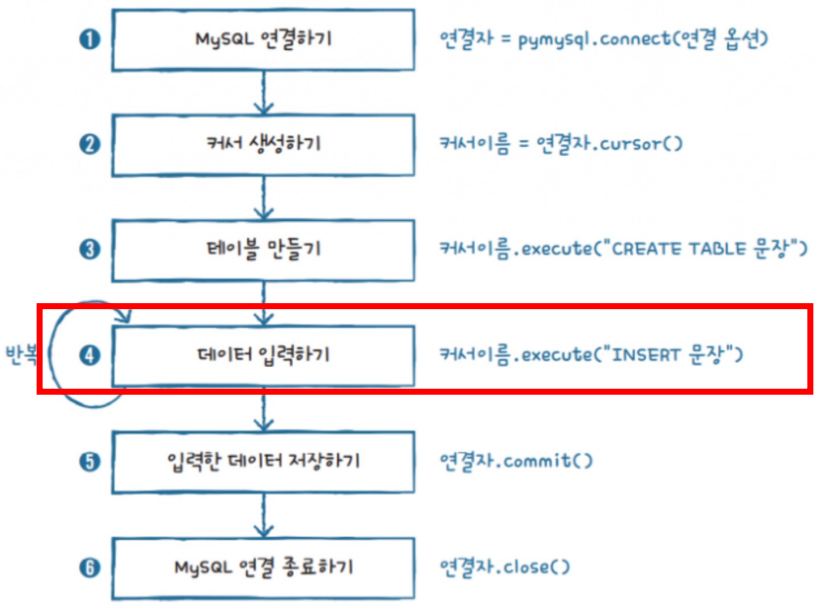



```python
if __name__ == '__main__':
    print('MySQL db test start')
    print('------------------------------------------------')
    # Mysql 연결
    conn = mysql_connect( ip: 'localhost', user: 'dbtest', pwd: 'password', db: 'dbtest')
    cur = conn.cursor()

    # MySQL Set Encryption mode
    query = "SELECT @@block_encryption_mode;"
    cur.execute(query)
    for cur_str in cur:
        if cur_str[0] != 'aes-256-cbc':
            if key_length==128:
                query = "SET block_encryption_mode='aes-128-cbc'"
                cur.execute(query)
                print("암호화 방식 'AES-128-CBC'으로 변경", end="\n\n")
            else:
                query = "SET block_encryption_mode='aes-256-cbc'"
                cur.execute(query)
                print("암호화 방식 'AES-256-CBC'으로 변경", end="\n\n")
```

```python
# Mysql 테이블 생성
print("MySQL 테이블 생성 실행")
try:
    cur.execute(table_init)
    cur.execute(table_key)
except Exception as ex:
    if ex.args[0] == 1050:
        query = "DROP TABLE test, key_manage"
        cur.execute(query)
        cur.execute(table_init)
        cur.execute(table_key)
print("MySQL 테이블 생성 성공", end="\n\n")
```

3. 구현 코드 분석




```python
# Bench Test using MySQL Function
mysql_bench('write_only')
mysql_bench('read_only')

# Database 초기화
print("Database 초기화중.....")
try:
    query = "DROP TABLE test, key_manage"
    cur.execute(query)
    cur.execute(table_init)
    cur.execute(table_key)
except Exception as ex:
    print(ex)
print("Database 초기화 완료", end="\n\n")

# Bench Test using Python library
python_bench('write_only')
python_bench('read_only')
```

3. 구현 코드 분석 - MySQL 함수 사용

```python
def mysql_write_only_test():
    print("MySQL 데이터 쓰기 성능 테스트 실행-----------------")
    time_=0
    for i in range(Iteration):
        try:
            mysql_set_randombyte()
            mysql_insert_key_table()
            query = ("INSERT INTO test(ciphertext_1, ciphertext_2, ciphertext_3) VALUES("
                     "HEX(AES_ENCRYPT(@text_1, @key, @iv)),"
                     "HEX(AES_ENCRYPT(@text_2, @key, @iv)),"
                     "HEX(AES_ENCRYPT(@text_3, @key, @iv))"
                     ")")
            start = time.time()
            cur.execute(query)
            end = time.time()
            time_ = time_ + (end - start)
        except Exception as ex:
            print(ex)
    print("WorkingTime: {} sec".format(time_))
    print("MySQL 데이터 쓰기 성능 테스트 종료-----------------", end="\n\n")
```

```python
def mysql_set_randombyte():
    length_1 = random.randint( a: 1, b: 7)
    length_3 = random.randint( a: 16, b: 512)
    cur.execute("SET @iv = HEX(RANDOM_BYTES(8))")
    cur.execute("SET @key = HEX(RANDOM_BYTES(16))")
    query = "SET @text_1 = HEX(RANDOM_BYTES(" + str(length_1) + "))"
    cur.execute(query)
    cur.execute("SET @text_2 = HEX(RANDOM_BYTES(16))")
    query = "SET @text_3 = HEX(RANDOM_BYTES(" + str(length_3) + "))"
    cur.execute(query)


1개의 사용 위치
def mysql_insert_key_table():
    cur.execute("INSERT INTO key_manage(temp_text_1,temp_text_2,temp_text_3, user_key, user_iv) "
                "VALUES(@text_1,@text_2,@text_3, @key, @iv)")
```

3. 구현 코드 분석 - MySQL 함수 사용

```python
def mysql_read_only_test():
    print("MySQL 데이터 읽기 성능 테스트 실행----------")
    total_1 = 0
    total_2 = 0
# 순서대로 값을 읽을 때
    for i in range(1, Iteration+1):
        start_1 = time.time()
        mysql_set_key_iv_cipher(i)
        end_1 = time.time()
        start_2 = time.time()
        cur.execute("SELECT AES_DECRYPT(UNHEX(@cipher_1), @key, @iv)")
        cur.execute("SELECT AES_DECRYPT(UNHEX(@cipher_2), @key, @iv)")
        cur.execute("SELECT AES_DECRYPT(UNHEX(@cipher_3), @key, @iv)")
        end_2 = time.time()
        total_1 = total_1 + (end_1 - start_1)
        total_2 = total_2 + (end_2 - start_2)
    print("In Order index WorkingTime")
    print("Set Key, Iv, Cipher : {}sec".format(total_1))
    print("Decrypt : {}sec".format(total_2))
```

```python
# Random한 값을 읽을 때
    random_index = mysql_random_index_init()
    for index in random_index:
        try:
            start_1 = time.time()
            mysql_set_key_iv_cipher(index)
            end_1 = time.time()
            start_2 = time.time()
            cur.execute("SELECT AES_DECRYPT(UNHEX(@cipher_1), @key, @iv)")
            cur.execute("SELECT AES_DECRYPT(UNHEX(@cipher_2), @key, @iv)")
            cur.execute("SELECT AES_DECRYPT(UNHEX(@cipher_3), @key, @iv)")
            end_2 = time.time()
            total_1 = total_1 + (end_1 - start_1)
            total_2 = total_2 + (end_2 - start_2)
        except Exception as ex:
            print(ex)
    print("Random index WorkingTime")
    print("Set Key, Iv, Cipher : {}sec".format(total_1))
    print("Decrypt : {}sec".format(total_2))

    print("MySQL 데이터 읽기 성능 테스트 종료----------", end="\n\n")
```

```python
def mysql_random_index_init():
    index = []
    for i in range(Iteration):
        index.append(random.randint( a: 1, Iteration))
    return index


2개의 사용 위치
def mysql_set_key_iv_cipher(index):
    cur.execute("SELECT @key := user_key FROM key_manage WHERE userid = " + str(index))
    cur.execute("SELECT @iv := user_iv FROM key_manage WHERE userid = " + str(index))
    cur.execute("SELECT @cipher_1 := ciphertext_1 FROM test WHERE userid = " + str(index))
    cur.execute("SELECT @cipher_2 := ciphertext_2 FROM test WHERE userid = " + str(index))
    cur.execute("SELECT @cipher_3 := ciphertext_3 FROM test WHERE userid = " + str(index))
```

3. 구현 코드 분석 - Python 라이브러리 사용

```python
def python_mysql_write_only_test():
    print("Python 암호화 함수 MySQL 데이터 쓰기 성능 테스트 실행----------------")
    time_ = 0
    for i in range(Iteration):
        try:
            key, iv, text_1, text_2, text_3 = python_mysql_set_randombyte(key_length)
            python_mysql_insert_key_table(key, iv, text_1, text_2, text_3)

            start = time.time()
            cipher = AES.new(key, mode, iv)
            text_1, text_2, text_3 = padding_data(text_1, text_2, text_3, pad)
            cipher_1 = cipher.encrypt(text_1)
            cipher_2 = cipher.encrypt(text_2)
            cipher_3 = cipher.encrypt(text_3)
            query = ("INSERT INTO test(ciphertext_1, ciphertext_2, ciphertext_3) VALUES ("
                     "'"+ cipher_1.hex() +"', "
                     "'"+ cipher_2.hex() +"', "
                     "'"+ cipher_3.hex() +"'"
                     ")")
            cur.execute(query)
            end = time.time()
            time_ = time_ + (end - start)
        except Exception as ex:
            print(ex)
    print("WorkingTime: {} sec".format(time_))
    print("Python 암호화 함수 MySQL 데이터 쓰기 성능 테스트 종료----------------", end="\n\n")
```

```python
def python_mysql_set_randombyte(length):
    if length==128:
        key = secrets.token_bytes(16)  # 16 bytes for AES-128
    else:
        key = secrets.token_bytes(32)  # 32 bytes for AES-256
    iv = secrets.token_bytes(16)
    length_1 = random.randint( a: 1,  b: 7)
    length_3 = random.randint( a: 16,  b: 512)
    data_1 = random.randbytes(length_1)
    data_2 = random.randbytes(16)
    data_3 = random.randbytes(length_3)
    return key, iv, data_1, data_2, data_3


1개의 사용 위치
def python_mysql_insert_key_table(key, iv, text_1, text_2, text_3):
    key_array.append(key)
    iv_array.append(iv)
    cur.execute("INSERT INTO key_manage(temp_text_1,temp_text_2,temp_text_3) "
                "VALUES("
                "'"+text_1.hex()+"', "
                "'"+text_2.hex()+"', "
                "'"+text_3.hex()+"')")


3개의 사용 위치
def padding_data(text1, text2, text3, func):
    pad_text1 = func(text1, AES.block_size)
    pad_text2 = func(text2, AES.block_size)
    pad_text3 = func(text3, AES.block_size)
    return pad_text1, pad_text2, pad_text3
```

3. 구현 코드 분석 - Python 라이브러리 사용

```python
# 순서대로 값을 읽을 때
    for i in range(1, Iteration+1):
        # 복호화를 위한 세팅
        start_1 = time.time()
        cur.execute("SELECT ciphertext_1, ciphertext_2, ciphertext_3 FROM test WHERE userid = " + str(i))
        for cur_str in cur:
            cipher_1 = bytes.fromhex(cur_str[0])
            cipher_2 = bytes.fromhex(cur_str[1])
            cipher_3 = bytes.fromhex(cur_str[2])
        key = key_array[i-1]
        iv = iv_array[i-1]
        end_1 = time.time()
        start_2 = time.time()
        cipher = AES.new(key, mode, iv)
        text_1 = cipher.decrypt(cipher_1)
        text_2 = cipher.decrypt(cipher_2)
        text_3 = cipher.decrypt(cipher_3)
        text_1, text_2, text_3 = padding_data(text_1, text_2, text_3, unpad)
        end_2 = time.time()
        total_1 = total_1 + (end_1 - start_1)
        total_2 = total_2 + (end_2 - start_2)
    print("In Order index WorkingTime")
    print("Set Key, Iv, Cipher : {}sec".format(total_1))
    print("Decrypt : {}sec".format(total_2))
```

```python
# Random한 값을 읽을 때
    random_index = mysql_random_index_init()
    for index in random_index:
        start_1 = time.time()
        cur.execute("SELECT ciphertext_1, ciphertext_2, ciphertext_3 FROM test WHERE userid = " + str(index))
        for cur_str in cur:
            cipher_1 = bytes.fromhex(cur_str[0])
            cipher_2 = bytes.fromhex(cur_str[1])
            cipher_3 = bytes.fromhex(cur_str[2])
        key = key_array[index - 1]
        iv = iv_array[index - 1]
        end_1 = time.time()
        start_2 = time.time()
        cipher = AES.new(key, mode, iv)
        text_1 = cipher.decrypt(cipher_1)
        text_2 = cipher.decrypt(cipher_2)
        text_3 = cipher.decrypt(cipher_3)
        text_1, text_2, text_3 = padding_data(text_1, text_2, text_3, unpad)
        end_2 = time.time()
        total_1 = total_1 + (end_1 - start_1)
        total_2 = total_2 + (end_2 - start_2)
    print("Random index WorkingTime")
    print("Set Key, Iv, Cipher : {}sec".format(total_1))
    print("Decrypt : {}sec".format(total_2))
```

4. 실행 결과

```
Iteration = 100000
key_length = 256
mode = AES.MODE_CBC
```

```
암호화 방식 'AES-256-CBC'으로 변경

MySQL 테이블 생성 실행
MySQL 테이블 생성 성공
```

```
MySQL 데이터 쓰기 성능 테스트 실행----------------
WorkingTime: 5.482355356216431 sec
MySQL 데이터 쓰기 성능 테스트 종료----------------

MySQL 데이터 읽기 성능 테스트 실행----------------
In Order index WorkingTime
Set Key, Iv, Cipher : 27.21866273880005sec
Decrypt : 14.5891871452331 54sec

Random index WorkingTime
Set Key, Iv, Cipher : 27.096022605895996sec
Decrypt : 13.94845700263977sec
MySQL 데이터 읽기 성능 테스트 종료----------------
```

```
Python 암호화 함수 MySQL 데이터 쓰기 성능 테스트 실행----------------
WorkingTime: 6.042705774307251 sec
Python 암호화 함수 MySQL 데이터 쓰기 성능 테스트 종료--------+--------

Python 암호화 함수 MySQL 데이터 읽기 성능 테스트 실행----------------
In Order index WorkingTime
Set Key, Iv, Cipher : 7.1877562999725 34sec
Decrypt : 1.3402070999145508sec

Random index WorkingTime
Set Key, Iv, Cipher : 7.303342580795288sec
Decrypt : 1.3559820652008057sec
Python 암호화 함수 MySQL 데이터 읽기 성능 테스트 종료----------------
```

4. 실행 결과

```
Iteration = 100000
key_length = 128
mode = AES.MODE_CBC
```

```
MySQL 데이터 쓰기 성능 테스트 실행----------------
WorkingTime: 5.701258420944214 sec
MySQL 데이터 쓰기 성능 테스트 종료----------------

MySQL 데이터 읽기 성능 테스트 실행----------------
In Order index WorkingTime
Set Key, Iv, Cipher : 27.348154306411743sec
Decrypt : 14.606437683105469sec

Random index WorkingTime
Set Key, Iv, Cipher : 27.08414936065674sec
Decrypt : 13.943898677825928sec
MySQL 데이터 읽기 성능 테스트 종료----------------
```

```
Python 암호화 함수 MySQL 데이터 쓰기 성능 테스트 실행----------------
WorkingTime: 6.027552127838135 sec
Python 암호화 함수 MySQL 데이터 쓰기 성능 테스트 종료----------------

Python 암호화 함수 MySQL 데이터 읽기 성능 테스트 실행----------------
In Order index WorkingTime
Set Key, Iv, Cipher : 7.17487430572509sec
Decrypt : 1.27689909934997sec

Random index WorkingTime
Set Key, Iv, Cipher : 7.2455902099609375sec
Decrypt : 1.2837498188018799sec
Python 암호화 함수 MySQL 데이터 읽기 성능 테스트 종료----------------
```

4. 실행 결과

	128	256
MySQL 입력	5.7012 s	5.4823 s
MySQL 읽기 – 순서대로 Set key, iv, cipher	27.3481 s	27.2186 s
MySQL 읽기 - 순서대로 Decrypt	14.6064 s	14.5891 s
MySQL 읽기 – 랜덤 Set key, iv, cipher	27.0841 s	27.0960 s
MySQL 읽기 - 랜덤 Decrypt	13.9438 s	13.9484 s
Python 함수 입력	6.0275 s	6.0427 s
Python 함수 읽기 - 순서대로 Set key, iv, cipher	7.1748 s	7.1877 s
Python 함수 읽기 - 순서대로 Decrypt	1.2768 s	1.3402 s
Python 함수 읽기 - 랜덤 Set key, iv, cipher	7.2455 s	7.3033 s
Python 함수 읽기 - 랜덤 Decrypt	1.2837 s	1.3559 s

5. 이후 진행

- 키 생성 및 키 변경 시 발생하는 보안 관련 조사
  - PQC로 키 생성 과정을 통해 신규키를 생성한다?
  - SGX를 통해 안전한 환경에서 암호화한다?
  - 동형암호로 암호화한다?
  - re-encryption 기법을 활용한다?

- 위와 관련된 자료를 조사하고 DB 시나리오에 적용할 때 어떻게 진행되어야 하는지 생각

Q & A