Шифрование хранимых данных в ClickHouse

Шифрование хранимых данных (Data Encryption at Rest)

- Защита данных на дисках от несанкционированного доступа, в том числе в случае утери или кражи диска

- Шифрование хранимых данных НЕ обеспечивает защиту данных, пересылаемых по сети. Но это может быть достигнуто другими средствами (см. *https_port*, *tcp_port_secure*, *interserver_https_port* и др.)

Способы шифрования данных:

Функции encrypt и decrypt

```
INSERT INTO mytable VALUES
encrypt('aes-128-cbc', 'plaintext',
'secretkey0123456')

SELECT decrypt('aes-128-cbc', x,
'secretkey0123456') FROM mytable
```

Зашифрованный виртуальный диск

new!

```
<clickhouse>
   <storage_configuration>
      <disks>
         <encrypted_disk>
            <type>encrypted</type>
            ...
```

Шифрующий кодек

new!

```
CODEC(AES_128_GCM_SIV)
CODEC(AES_256_GCM_SIV)
```

Виртуальные диски

```
CREATE TABLE mytable (x String) ENGINE=MergeTree ORDER BY tuple()
```

- данные сохраняются в папке `/var/lib/clickhouse/`

конфигурация:

```
<clickhouse>
    <storage_configuration>
        <disks>
            <mydisk>
                <type>local</type>
                <path>/mydisk/</path>
            </mydisk>
        </disks>
        <policies>
            <mypolicy>
                <volumes>
                    <main>
                        <disk>mydisk</disk>
                    </main>
                </volumes>
            </mypolicy>
        </policies>
    </storage_configuration>
</clickhouse>
```

```
CREATE TABLE mytable (x String) ENGINE=MergeTree ORDER BY tuple()
SETTINGS storage_policy='mypolicy'
```

- данные сохраняются в папке `/mydisk/`

Виртуальные диски


```
<disks>
   <mydisk>
      <type>local</type>
      <path>/mydisk/</path>
   </mydisk>
</disks>
```

Тип диска указывает, каким образом диск хранит данные.

Поддерживаемые типы дисков: local, memory, s3, hdfs, web, encrypted

Зашифрованный виртуальный диск: пример

Конфигурация:

```
<clickhouse>
   <storage_configuration>
      <disks>
         <local_disk>
            <type>local</type>
            <path>/disk/</path>
         </local_disk>
         <encrypted_disk>
            <type>encrypted</type>
            <disk>local_disk</disk>
            <path>encrypted/</path>
            <key>secretkey0123456</key>
         </encrypted_disk>
      </disks>
      <policies>
         <encrypted_policy>
            <volumes>
               <main>
                  <disk>encrypted_disk</disk>
               </main>
            </volumes>
         </encrypted_policy>
      </policies>
   </storage_configuration>
</clickhouse>
```

авторы реализации:
Александра Латышева
Виталий Баранов

Зашифрованный виртуальный диск: пример


```
CREATE TABLE mytable (x String) ENGINE=MergeTree ORDER BY tuple()
SETTINGS storage_policy='encrypted_policy';
```

```
INSERT INTO mytable VALUES ('plaintext');

INSERT INTO mytable VALUES ('plaintext2');

SELECT * FROM mytable;
```

```
┌─x──────────┐
│ plaintext  │
└────────────┘

┌─x──────────┐
│ plaintext2 │
└────────────┘

2 rows in set. Elapsed: 0.003 sec.
```

Вставка в таблицу на зашифрованном диске

Представление данных на зашифрованном диске

/disk/encrypted/store/751/751f7fee-3e58-4782-b51f-7fee3e588782/

```
all_1_1_0/
    checksums.txt
    columns.txt
    count.txt
    default_compression_codec.txt
    data.bin
    data.mrk3
```

```
all_2_2_0/
    checksums.txt
    columns.txt
    count.txt
    default_compression_codec.txt
    data.bin
    data.mrk3
```

```
detached
```

```
format_version.txt
```

```
$ hexdump -C all_1_1_0\data.bin
00000000  45 4e 43 01 00 00 00 00  00 00 00 00 00 00 00 03  |ENC.............|
00000010  d8 5d a6 56 1d 67 ab c5  a7 e4 2c b1 bd f6 f4 6f  |.].V.g....,....o|
```

Сигнатура файла

Ключи шифрования

1) Ключи можно (и рекомендуется) задавать в шестнадцатеричном виде:

```
<key_hex>00112233445566778899aabbccddeeff</key_hex>
```

2) Длина ключа зависит от алгоритма шифрования:

```
<encrypted_disk>
   <type>encrypted</type>
   <disk>local_disk</disk>
   <algorithm>aes_128_ctr</algorithm>
   <key>1234567890123456</key>
</encrypted_disk>
```

aes_128_ctr - алгоритм по-умолчанию

```
<algorithm>aes_192_ctr</algorithm>
<key>123456789012345678901234</key>
```

```
<algorithm>aes_256_ctr</algorithm>
<key>12345678901234567890123456789012</key>
```

Ключи шифрования

3) Хранить ключи прямо в основном файле конфигурации **небезопасно**.

Альтернативы:

Передача ключа через переменную окружения

```
<disks>
   <encrypted_disk>
      <key from_env="ENCKEY"/>
      ...
   </encrypted_disk>
</disks>
```

```
$ ENCKEY=secretkey0123456
/usr/bin/clickhouse-server
```

Символическая ссылка в /etc/clickhouse-server/config.d/

```
$ ln -s /media/usb/keys.xml
/etc/clickhouse-server/config.d/keys.xml
```

/media/usb/keys.xml

```
<clickhouse>
   <storage_configuration>
      <disks>
         <encrypted_disk>
            <key>secretkey0123456</key>
         </encrypted_disk>
      </disks>
   </storage_configuration>
</clickhouse>
```

Ключи шифрования

4) Можно одновременно использовать много ключей.

```
<disks>
   <encrypted_disk>
      <type>encrypted</type>
      <disk>local_disk</disk>
      <path>encrypted/</path>
      <key id="0">zerokeyzerokey__</key>
      <key id="1">firstkeyfirstkey</key>
      <key id="2">secondkeysecond_</key>
      <current_key_id>2</current_key_id>
   </encrypted_disk>
</disks>
```

При записи новых данных используется текущий ключ (*current_key_id*)

При чтении данных могут использоваться любые ключи (не только *current_key_id*)

```
$ hexdump -C all_1_1_0\data.bin
00000000  45 4e 43 01 00 00 00 02  00 00 00 00 00 00 00 0c  |ENC.............|
00000010  e9 ae 08 b1 96 9b d7 40  a9 6b 50 92 c0 ba 91 bc  |.......@.kP.....|
```

ID ключа, необходимого для расшифровки

Ключи шифрования

Добавление нового ключа:

```
<disks>
    <encrypted_disk>
        <type>encrypted</type>
        <disk>local_disk</disk>
        <path>encrypted/</path>
        <key id="0">zerokeyzerokey__</key>
        <key id="1">firstkeyfirstkey</key>
        <key id="2">secondkeysecond_</key>
-       <current_key_id>2</current_key_id>
+       <key id="3">thirdsecretkey__</key>
+       <current_key_id>3</current_key_id>
    </encrypted_disk>
</disks>
```

Не рекомендуется:
модификация существующих ключей,
удаление старых ключей

Размещение зашифрованных файлов


```
<disks>
    <local_disk>
        <type>local</type>
        <path>/disk/</path>
    </local_disk>
    <encrypted_disk>
        <type>encrypted</type>
        <disk>local_disk</disk>
        <path>encrypted/</path>
        <key>secretkey0123456</key>
    </encrypted_disk>
</disks>
```

файлы будут размещены в /disk/encrypted/

```
<encrypted_disk>
    <type>encrypted</type>
    <disk>s3_disk</disk>
    <key>secretkey0123456</key>
</encrypted_disk>
```

в Amazon S3

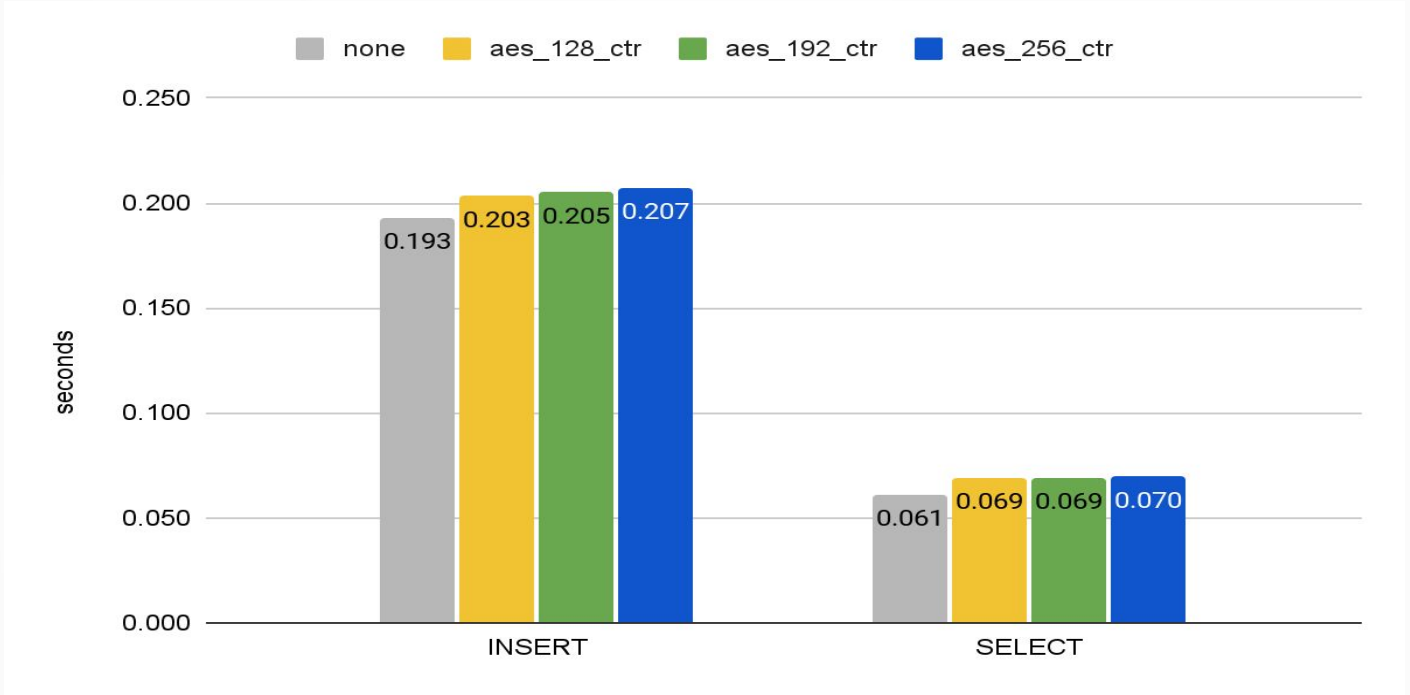
```
<encrypted_disk>
    <type>encrypted</type>
    <disk>hdfs_disk</disk>
    <key>secretkey0123456</key>
</encrypted_disk>
```

в HDFS

Зашифрованный диск: Примечания

1. шифрование происходит при записи на диск, расшифровка при чтении с диска
2. применимо для движков таблиц семейств MergeTree и Log
3. шифруются все данные, в том числе засечки и контрольные суммы
4. метадата (определения таблиц) не зашифрована
5. при чтении расшифровываются только необходимые данные
6. при слиянии кусков MergeTree выполняется расшифровка, слияние, и потом шифрование
7. при репликации в ReplicatedMergeTree данных кусок расшифровывается, пересылается, и потом шифруется на другой реплике снова (см. interserver_https_port !!!)
8. ключи на разных нодах могут не совпадать

Производительность зашифрованного диска




```
INSERT INTO mytable SELECT number FROM numbers(10000000);
SELECT sum(x) FROM mytable;
```

Зашифрованный диск: Алгоритм AES CTR

+ произвольный доступ (можно расшифровать данные из середины файла)
+ размер не изменяется (количество байт до и после шифрования одно и то же), сохраняются смещения
+ можно добавить данные в конец файла без расшифровки предыдущих данных
+ быстро работает
- iv не должен использоваться повторно (но мы это решили)

```
$ hexdump -C all_1_1_0\data.bin

00000000  45 4e 43 01 00 00 00 00  00 00 00 00 00 00 00 03  |ENC.............|

00000010  d8 5d a6 56 1d 67 ab c5  a7 e4 2c b1 bd f6 f4 6f  |.].V.g....,....o|
```

Случайно сгенерированный iv

Шифрование столбцов

```
<clickhouse>
    <encryption_codecs>
        <aes_128_gcm_siv>
            <key>0123456789abcdef</key>
        </aes_128_gcm_siv>
        <aes_256_gcm_siv>
             <key>abcdefghijklmnopabcdefghijklmnop</key>
        </aes_256_gcm_siv>
    </encryption_codecs>
</clickhouse>
```

```
CREATE TABLE mytable (x String Codec(AES_128_GCM_SIV))
ENGINE=MergeTree ORDER BY x;
```

Автор идеи:
depressed-pho

авторы реализации:
depressed-pho
Филатенков Артур

Пример работы

```
INSERT INTO mytable VALUES ('plaintext');

INSERT INTO mytable VALUES ('plaintext2');

SELECT * FROM mytable;
```

```
┌─x─────────┐
│ plaintext │
└───────────┘

┌─x──────────┐
│ plaintext2 │
└────────────┘

2 rows in set. Elapsed: 0.013 sec.
```

Порядок кодеков

```
CREATE TABLE mytable (x String Codec(Delta, LZ4, AES_128_GCM_SIV))
ENGINE=MergeTree ORDER BY x;
```

Порядок кодеков:

1. Специальные (Delta, DoubleDelta, Gorilla, T64)
2. Сжатие общего назначения (ZSTD, LZ4)
3. Шифрование (AES_128_GCM_SIV, AES_256_GCM_SIV)

Порядок кодеков

Шифрование без сжатия

```
CREATE TABLE mytable (x String Codec(AES_128_GCM_SIV) )
ENGINE=MergeTree ORDER BY x;
```

Сжатие и шифрование

```
CREATE TABLE mytable (x String, Codec(LZ4, AES_128_GCM_SIV) )
ENGINE=MergeTree ORDER BY x;
```

Подробнее о конфиге

```
<clickhouse>
    <encryption_codecs>
        <aes_128_gcm_siv>
            <key>0123456789abcdef</key>
        </aes_128_gcm_siv>
        <aes_256_gcm_siv>
             <key_hex id="0">00112233445566778899aabbccddeeff</key_hex>
             <key_hex id="1" from_env="ENCKEY"></key_hex>
             <current_key_id>1</current_key_id>
        </aes_256_gcm_siv>
    </encryption_codecs>
</clickhouse>
```

SIV Algorithms and Nonce

```
<clickhouse>
    <encryption_codecs>
        <aes_128_gcm_siv>
            <key>0123456789abcdef</key>
            <nonce>0123456789101</nonce>
        </aes_128_gcm_siv>
    </encryption_codecs>
</clickhouse>
```

- Алгоритмы шифрования обладают свойством nonce misuse resistance
- Значение по умолчанию:

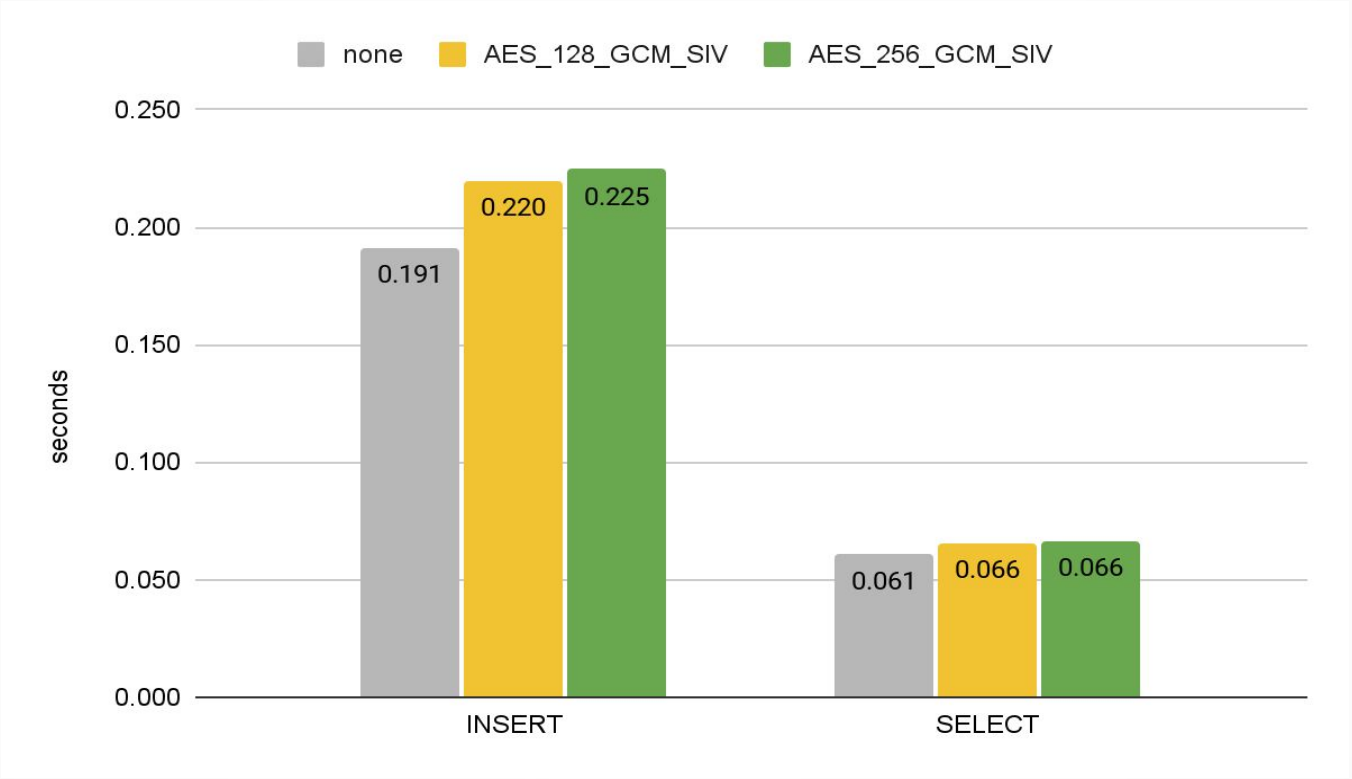
```
<nonce_hex>000000000000000000000000</nonce_hex>
```

Бинарное представление зашифрованных данных

- Key_id - id ключа
- Nonce - nonce, если он определен
- Encrypted_data - зашифрованные данные

...	key_id	nonce	encrypted_data

Производительность




```
INSERT INTO mytable SELECT number FROM numbers(10000000);
SELECT sum(x) FROM mytable;
```

Особенности шифрования с помощью кодека

- Возможность зашифровать только часть столбцов
- При слиянии кусков выполняется расшифровка, затем слияние, потом шифрование
- Для репликации данных необходимо иметь одинаковые ключи на узлах

Результаты

Зашифрованный виртуальный диск (21.9)

```
<encrypted_disk>
   <type>encrypted</type>
   <disk>local_disk</disk>
</encrypted_disk>
```

Шифрующий кодек (21.11)

```
CODEC(AES_128_GCM_SIV)
CODEC(AES_256_GCM_SIV)
```

Планы

- Интеграция с KMS (AWS KMS и др.)
- Улучшить управление ключами (переход на новый ключ вместо старого)
- Улучшить шифрующий кодек (генерация nonce, разные ключи для разных столбцов)
- Предлагайте!

Спасибо за внимание