# CryptDB实现按列选择加密

## 一、建表：create table test(id int,name varchar(50))

**思路**：在SQL语句中需要表示是否对该列进行加密，原始想法是在数据类型后增加一个flag标识为，然而这样需要需要mysql底层的SQL语句语法，很明显不太现实，没有可操作性。因为选择对列名进行一些处理，需要加密的列，列名前缀规定为“enc_”。在SQL语句改写时，如果读取到的列名是以“enc”开头的，就进行正常的加密，反之，明文。具体修改函数如下：

create语句是ddl，通过分配的ddl_handler（ddl_handler.cc) 进行处理，CreateTableHandler 是DDLHandler 的子类，其中 “ new_lex->alter_info.create_list =**accumList**<Create_field>(it, [&a, &ps, &tm] List<Create_field> out_list,Create_field *const cf) {return **createAndRewriteField**(a, ps, cf, tm.get(),true, out_list); }); ” 是改写的关键语句。对表中的每列，调用**createAndRewriteField**函数，创建FieldMeta并进行重写。

1. **rewrite_util.cc——createAndRewriteField**

```
/*对于每一列，调用createAndRewriteField，首先对列名进行提取，判断前缀，给flag赋值*/
const std::string name = std::string(cf->field_name);
std::cout <<"field_name is "<<name<<std::endl;
std::string temp = name.substr(0,4);
std::string encflag ("enc_");
bool flag = 0;
if (temp.compare(encflag)==0)
flag = 1;
```

```
/*rewrite_create_field函数增加一个参数flag，将列名中的加密特征传递到底层加密层*/
const auto new_fields = rewrite_create_field(fm.get(), cf, a, flag);
```

2. **rewrite_util.hh**

```
/*修改rewrite_create_field头文件*/
std::vector<Create_field *>
rewrite_create_field(const FieldMeta * const fm, Create_field * const f,
                     const Analysis &a, bool flag);
```

3. **rewrite_util.cc——rewrite_create_field**

```
/*如果flag为0，则不加密，参数中传递的原始Create_field *f 就是改写之后的field*/
if (flag == 0)
    output_cfields.push_back(f);
else
    {...}
```

**结果：**

create table test321(id int,enc_name varchar(20));

```
[1;31mQUERY: [0m INSERT INTO remote_db.generic_prefix_remoteQueryCompletion   (begin, complete, embedded_completion_id, reissue) VALUES   (TRUE,  FALSE,530    , FALSE);
[1;31mQUERY: [0mcreate table table_IEFNRDJCBA (id INT(11), GUJESBSUELoDET VARBINARY(32), YHRDGNXXVHoOPE BIGINT(20), cdb_saltRPOXCTHNAE BIGINT(8) unsigned not null) ENGINE=InnoDB
[1;31mQUERY: [0m UPDATE remote_db.generic_prefix_remoteQueryCompletion    SET complete = TRUE  WHERE embedded_completion_id = 530;
```

## 二、插入：insert into test values(1,'aaa');

**思路**：插入数据时会对表名和插入的数据进行重写，关注数据。思路和建表一样，找到列名和原始数据，在进行加密时，加一个判断。

insert语句属于dml语句，通过dml_handler处理 (dml_handler.cc)，dml处理有两个函数，**gather**和**rewrite**，gather函数应该是对需要加密的数据进行聚集，并且返回用什么算法加密，然而在调试时发现它好像并没有进入最核心的**gatherAndAddAnalysisRewritePlan**函数。那就看rewrite函数，在加密之前截断！InsertHandler 继承了 DMLHandler，其中for循环，对于每个传入的值调用了**rewriteInsertHelper**函数，进行加密，修改都在这个函数中。

1. **dml_handler.cc——rewriteInsertHelper**

```
/*fm的类型是FieldMeta，有一个成员变量fname，列名，可以直接获取。之后判断是否需要加密*/
std::string name = fm.fname;
std::cout<<"name is "<<name<<std::endl;
std::string temp = name.substr(0,4);
std::string encflag ("enc_");
bool flag = 0;
if (temp.compare(encflag)==0)
    flag = 1;
```

```
/*如果是1，正常加密，反之，将原始数据i写入列表中，这里需要注意push_back需要的参数是Item*类型，而i是const Item类型的，需要进行准换*/
if (flag == 1) {...}
else
{
    Item *p = const_cast<Item *>(&i);
    append_list->push_back(p);
 }
```

**结果**:

insert into test321 values(1,"aaa");

```
[1;31mQUERY: [0minsert into `xinan`.`table_IEFNRDJCBA` values (1, '1B4E7196529D62139422F9417639A725', 4702133347715229456, 18087396492744362312)
```

## 三、选择：select * from test

**思路**：通过调试发现，gather函数是在select语句中使用的，开始想的是只将需要加密的列返回，在进行后续的rewrite，但是不对，因为gather返回的是需要查询的列的信息，是包含数据库名和表名的，"xinan.test.id"和"xinan.test.enc_name"，如果不返回不加密的那个item，那SQL语句中表名也就不会被加密处理了，但是实际上虽然列不加密，但表名都是加密存储的，因此修改操作还是要集中在rewrite函数中。

rewrite_select_lex —— > rewrite_proj——> do_rewrite ——> do_rewrite_type

1. rewrite_field.cc——do_rewrite_type

```
/*对gather返回的item中的列名进行提取，然后判断，给flag赋值*/
const std::string name = i.field_name;;
std::cout<<"name is "<<name<<std::endl;
std::string temp = name.substr(0,4);
std::string encflag ("enc_");
bool flag = 0;
if (temp.compare(encflag)==0)
    flag = 1;
```

```
/*如果不加密，就把原始列名加入item_field就可以*/
if (flag == 1) {...}
else
    res = make_item_field(i, anon_table_name, name);
...
return res;
```

**结果**:

select * from test;

```
[1;31mQUERY: [0mselect `xinan`.`table_YMUOVNTOJZ`.`id`,`xinan`.`table_YMUOVNTOJZ`.`GRBGOPBZYNoDET` from `xinan`.`table_YMUOVNTOJZ`
```

select不是这么简单就完事了，还有where having之类的关键词需要处理。

但是在处理之前已经发现问题了，在3307端口，select的时候会报错。

```
mysql> select * from test3;
ERROR 1105 (07000): AES padding is wrong size!
```

感觉是因为只改了表面，没改里面？？ 3307和3306两个端口到底分别是什么，正确的明密文是什么样的？ 3307如果是客户端看到的透明的，那就是它解密的时候出问题？有点乱。

————————————————————2020.09.15 LYJ————————————————————