CryptDB数据库加密系统

CryptDB采用的加密策略有多种，可以分为6类：RND、DET、OPE、HOM、JOIN、Search，这些加密策略使用的加密算法包括成熟的加密算法比如AES、Paillier等，也包括为特定的操作符（Join）设计的加密算法，所有加密算法都是对称加密算法。

**一、加密算法**

（1）RND(Random)，是CryptDB提供的最安全的加密策略。RND的加密特征是两个相同的明文加密之后密文不同，因而不能有效地支持基于密文的任何计算。RND的实现算法主要采用分组密码AES或者Blowfish，算法采用CBC模式并且需要一个初始化向量IV。

（2）DET(Deterministic)，加密特征是相同的明文加密之后密文相同。采用这种加密策略会泄露部分信息给DBMS，因此相对来说安全性较弱。但是这种加密方式允许DBMS服务器基于加密数据执行等值检查计算，包括等值连接、Group by、Count、Distinct等操作。DET的实现采用分组密码算法AES，算法采用CMC模式并且不需要初始化向量IV。

（3）OPE(Order-preserving encryption)，称为保序加密，加密特征是明文的大小排序关系依然保留在密文中，也就是说: ，因此这种加密策略支持范围查询，也就是如果查询明文[c1,c2]内的数据可以等价地转换为查询密文范围内的数据，这种加密策略支持DBMS服务器执行Order by、min、max、sort等的查询。OPE的实现算法是Boldyreva等在2009年提出的算法。

（4）HOM(Homomorphic encryption，同态加密)：该算法允许服务器基于加密数据进行计算而最终的结果由CryptDB proxy进行解密。虽然全同态加密由于性能原因不可取，但是对于有限的几个操作实现全同态加密是可行的。例如

，为了计算，该UDF执行，解密之后就是的值。通过DBMS服务器分别返回几个数的和以及参与运算数据的数目。HOM还能够用来计算平均值，但是对于变量值的自增长比如set id=id+1，目前还不支持。HOM的实现算法是Pailler算法。

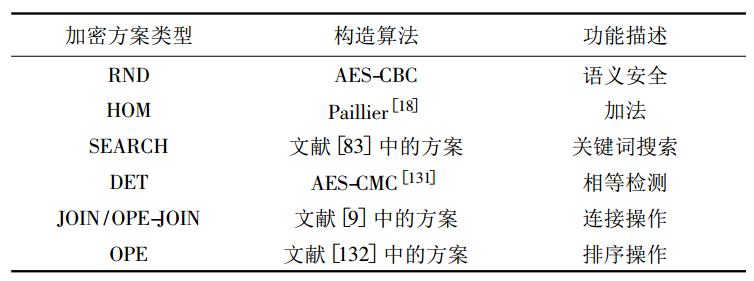
（5）JOIN(JOIN和OPE-JOIN)，在两列之间执行等值连接需要一个单独的加密算法，原因是如果采用DET加密策略，为了防止获知列之间的关系，不同的列之间加密密钥是不同的，因此不能直接基于DET加密策略执行连接。JOIN加密策略除了适用于连接操作之外，对于所有DET所支持的查询操作也都支持，而OPE-JOIN支持不等值连接。

（6）Search:这种机密模式是用来支持SQL中的Like查询的。Search只支持关键字搜索，不支持正则表达式匹配。

**二、洋葱加密策略**

如果应用程序对某列数据的查询不涉及比较和排序操作，考虑到隐私保护，该列应该用安全性最强的RND加密策略；而当查询涉及到等职比较但是不要求排序时，采用DET就够了。但是查询并不总是能提前知道，所以需要一个自适应调整方案，CryptDB可以基于查询动态调整DBMS中的数据加密策略，将加密策略调整到能够运行所请求的查询的最安全的加密策略。

a) 基于所有的 SQL 语句都可以 分 解 为 相 等 检 测 ( equality checks) 、 顺 序 比 较 ( order comparison) 、 聚 合 ( aggregate) 、 连 接 ( join) 四 种 基 本 操 作，CryptDB 根据应用需求用支持这四种操作的 SQL 可识别加密( SQL-aware encryption) 算法对数据库的每一列进行加密， 由于CryptDB 采用的大多数加密方案为对称加密， 所以系统效率影响相对较小; b) 由于支持不同功能的加密方案的安全性不一致，为了提供尽可能高的安全性， CryptDB 采用洋葱加密方式( onions of encryption) 。安全等级较低但功能较强的加密方案  
接近洋葱芯，而满足语义安全的加密方式则在最外层。通过引入用户定义函数( user defined functions，UDF) ，DBMS 软件本身不需要进行修改。在初始阶段数据被以洋葱加密的方式层层加密直至最外层达到语义安全，而在后续系统运行阶段，CryptDB 代理根据用户的查询请求进行拨洋葱操作。



在CryptDB系统中，每一个数据嵌入到多个加密圈(加密圈由多个加密策略层组成，越往外层加密策略的安全强度越高，也形象地成为“洋葱加密圈”)，每一种加密圈支持一类操作，通常有四种加密圈，如图1所示，其中Eq加密圈支持等值比较操作，包括等值选择和等值连接操作，Ord加密圈支持排序操作，Search加密圈支持字符串型数据的关键字匹配操作，该加密层对于整数列没有意义，Add加密圈支持数值型数据的求和计算，对于字符串列没有意义的。图二的Employees表包括两列，其中ID列定义为整型，而name列定义为字符串型。因此对于ID列的值，需要分别记录其对应的Eq、Ord和Add加密层的值，以及RND加密策略所需要的初始化向量IV。而Name列的值需要分别记录其对应的Eq、Ord和Search加密层的值以及RND加密策略所需要的初始化向量IV。这样就完成了原始Employees表的加密。这种加密方式不仅对数据进行了隐藏，而且加密后列的数量、表名都发生了变化，进一步保护了隐私数据。

用户发出查询请求时， CryptDB 工作流程如下: 应用服务器转发该查询请求到代理， 代理进行解释和重写: 匿名化表名称和列名称，根据查询中包含的操作使用系统主密钥加密查询中包含的常数; 代理检查 DBMS 服务器在执行查询之前是否需要调整加密层次， 如果需要调整则 发 送 一 个 update 命 令 给DBMS，之后 DBMS 通过调用 UDF 去调整加密层次; 代理将加密的查询请求发送给 DBMS 服务器，接到请求后 DBMS 服务器运行标准 SQL 查询( 在加和操作和关键词查询时调用 UDF) ;DBMS 服务器返回加密查询结果，代理解密后返回给应用服务器，应用服务器返回给发送查询请求的用户。

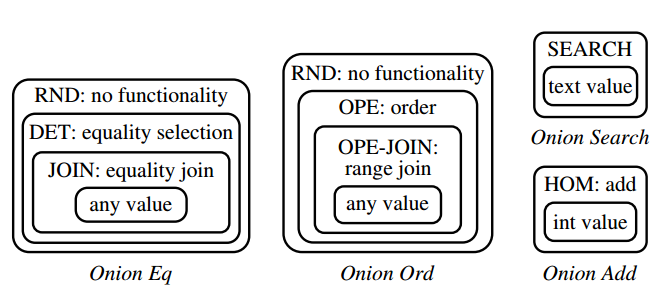


图1 洋葱加密圈

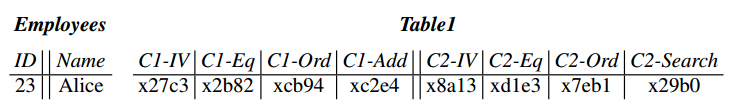


图2 加密示例

CryptDB对于表中同一列的值在相同的加密层上使用相同的密钥加密，而不同的表、列、加密圈、加密层则采用不同的加密密钥。而这些加密密钥来自于主键MK。例如，对于表t、列c、洋葱o和加密层l，使用的密钥：



其中，PRP是一个伪随机序列，主密钥保存在JDBC或者ODBC客户端，在对托管数据加密时将其传递给CryptDB，CryptDB根据上述公式获得加解密的密钥。

每个加密圈最外层采用最安全的加密方案，在出事情况下，所有的数据一致加密到最外层，比如Eq加密圈加密过程为数据依次利用JOIN加密策略、DET加密策略和RND加密策略得到加密之后的结果。当收到SQL查询时，CryptDB根据查询类型决定是否需要剥除当前加密层，例如如果一个查询需要在密文上执行谓词P，那么CryptDB将需要确定利用哪个加密圈的哪个加密层来执行P，如果c当前不在合适的加密层，则剥除一些加密层直到允许P。由于采用的加密策略都是对称加密算法，因此剥除加密层实际上就是对数据进行解密。CryptDB通过在DBMS服务器上执行UDF实现加密层的解密，例如在图2中，为了将第二列从Ord加密圈的的RND层剥除到OPE层，CryptDB proxy提交以下查询到服务器，查询中的DECRYPT\_RND为UDF，K是密钥生成公式计算得到的密钥。

UPDATE Tables

SET C2-Ord=DECRYPT\_RND(K,C2-Ord，C2-IV)

Write query execution写请求的操作语句：

为了支持INSERT, DELETE, 和UPDATE 查询语句，代理应用

**三、基于加密数据的查询示例**

下面以图二为例说明系统SQL语句查询的执行过程，初始情况下，DBMS的数据处于各种加密圈的最外层，当应用系统提交如下查询：

SELECT ID

FROM Employees

WHERE Name=’Alice’;

查询首先由CryptDB proxy进行处理，由于查询涉及到等值比较，需要将查询条件中的属性Name对应的Eq加密圈的加密层从RND降为DET层。因此，CryptDB proxy首先向DBMS提交如下修改语句：

UPDATE Table1

SET C2-Eq=

DECRYPT\_RND(KT1,C2,Eq,RND,C2-Eq,C2-IV);

接下来提交SQL查询：

SELECT C1-Eq,C1-IV

FROM Table1

WHERE C2-Eq=x7..d;

这里C1对应列ID,x7..d是’Alice’的密文，查询结果需要同时获得IV向量以最终能解密。这样，当CryptDB proxy 接收到查询结果后，根据保存的各个加密圈当前的状态信息，利用合适的解密算法和解密密钥就能获得查询结果的明文信息。