

数据库加密与验证机制的研究

邢国正,江雨燕,李 洁

(安徽工业大学 管理科学与工程学院,安徽 马鞍山 243002)

摘 要:以关系数据库为数据模型,在分析数种已有的数据加密技术基础上,研究数据库数据以密文方式存储并建立相应的数据库验证机制,立足于解决数据库中最为关键的数据验证机制等方面的理论、方法和技术问题;提出基于连接的子密钥生成算法以及基于用户信息的密钥管理技术和字段的记录验证技术,保证数据库的安全;有效地管理生成子密钥的字段加密技术所需的密钥,适应于客户端或服务端的加解密,适应于字段验证技术的灵活性和记录验证技术的安全性。

关键词:数据库;中国剩余定理;加密技术;验证技术

中图分类号:TP393.08

文献标识码:A

文章编号:1673-629X(2008)12-0166-04

Research on Database Encryption and Confirmation Mechanism

XING Guo-zheng,JIANG Yu-yan,LI Jie

(College of Management Sci. & Tech., Anhui Univ. of Tech., Maanshan 243002, China)

Abstract: Taking the relational database as the data model, based on analyzing several kinds already in the data encryption technology which has, studies the database data to save and to establish the corresponding database confirmation mechanism by the scrambled text way. Based on the database, solve the most critical database authentication mechanisms such as the theories, methods and technical issues. Proposes based on the connection sub-key production algorithm as well as based on the user information key management technology and the field record confirmation technology, guarantees the database the security. Effective management of the generation of key fields for key encryption technology, adapted to the client or the server encryption, authentication technology adapted to the flexibility of field records and verify the security of technology.

Key words: database; Chinese remainder theorem; encryption technology; confirmation technology

1 数据库安全问题的提出

数据库加密系统能够对数据库中的明文数据进行加密,并以密文保存;当合法用户读取数据时,该系统能够将密文解密成明文返回给用户。通过数据库加密可以使数据库具备机密性,数据库验证系统能够检查数据库中的数据是否被修改,并在必要时可以恢复数据库中的全部或部分数据。通过数据库验证可以使数据库具备完整性,数据库完整性主要包括以下三个方面:语义完整性、结构完整性、数据完整性。前两层的完整性维护在数据库管理系统中一般都已经有了完整的实现,但是目前数据完整性方面的研究还处于起步阶段^[1]。

安全的数据库系统具备机密性、完整性与可用性

三项基本条件。机密性是为了保护敏感性的数据避免被非法用户窃知;完整性是要防范数据库被有意或无意的破坏,以维持数据的正确性;而可用性是一旦数据库遭受不当的修改时,应能迅速恢复正常运作的能力,如复原或备份。三项基本条件中,机密性约束数据访问者,只有合法用户能够访问数据库中机密信息;完整性和可用性约束数据修改者,只有合法用户能够对数据库中机密信息进行合法修改。为了保护数据库的安全,人们做了大量的研究,主要可以分为三个层次来实现数据库安全,如图1所示。

上述前两个层次上的数据库安全研究工作已经充分开展,目前多数数据库应用系统都建立了基于前两个层次的安全系统,例如设置防火墙和分布式入侵检测系统、采用口令、访问权控制等等。这的确可以在一定程度上遏制黑客的入侵,但每周都有新的系统漏洞发现,手段高明的入侵者和新的攻击手法仍然能得逞。美国安全杂志“SECURE CYBERSPACE”调查,89%的用户安装了防火墙,60%的用户安装了入侵检测系统,但其中仍有90%的用户的系统安全受到破坏。所

收稿日期:2008-03-09

基金项目:安徽省自然科学基金项目(2005kj074)

作者简介:邢国正(1977-),男,安徽马鞍山人,讲师,硕士,研究方向为网络工程、数据库理论与应用;江雨燕,副教授,研究方向为MIS信息系统、CSCW、数据库理论与应用。

以,网络安全措施不是信息安全的全部,来自外部的黑客和来自内部的攻击行为不可避免,而且这些非法攻击的真正目标是数据库。即使系统从网络角度安装了相关设备或软件,从应用软件角度设置了访问控制和权限设置,仍然无法杜绝非法的外部访问,内部人员直接对数据表的修改更是难以被发现。只有对数据实施保护并建立验证机制,才是数据库安全的真正根本有效方法。目前无论是政府机构还是企业都开发了大量基于数据库的应用系统,但在实际项目开发和调研中,数据库几乎都是明文存储,没有任何加密措施。不通过应用程序和 DBMS 直接修改数据表的情况常常发生。因此数据库存储安全已经成为信息系统安全的核心,为了保护存储在数据库中的数据(信息)的安全,开发支持数据库加密与验证的软件,对目前信息系统应用具有十分重要的迫切性和现实意义。

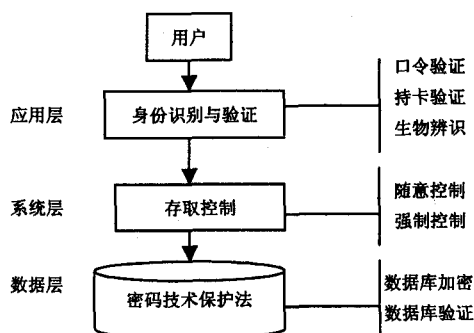


图 1 数据库安全研究层次

2 数据库加密与解密机制的研究

2.1 数据库加密粒度研究和实现

与一般的加密技术不同,由于数据库具有文件(表)、记录、字段多个层次的概念,因此对数据库的加密也可以分别选用文件(表)、记录、字段作为加密基本单位(加密粒度)的技术^[2]。加密的单位越小,适用的范围越广,但实现的难度也越大。对数据库加密粒度的选择可以有以下几种:

(1) 基于文件的数据库加密技术:把数据库文件作为整体,用加密密钥和加密算法对整个数据库文件加密,形成密文来保护数据的机密性^[3]。

(2) 基于记录的数据库加密技术:一般而言,数据库系统中每条记录所包含的信息具有一定的封闭性,即从某种程度上说它独立完整地存储了一个实体的数据。因此,基于记录的加密技术是常用的数据库信息加密手段。

(3) 子密钥加密技术:为了解决基于记录的数据库加密技术存在的问题,可以考虑采用一种子密钥数据库加密技术。该加密算法的核心思想是根据数据库

(特别是关系型数据库)中数据组织的特点,在加密时以记录为单位进行加密操作,而在解密时以字段为单位对单项数据进行解密操作。两者所用的密钥是不同的,加密所用的密钥是针对整个记录的密钥,而解密所用的密钥是针对单个数据项的子密钥。该算法的理论依据是著名的中国剩余定理。

(4) 基于字段的加密技术:基于字段的数据库加密,就是以不同记录的不同字段为基本加密单元进行加密。该方法可以对数据库中单个数据元素进行加密。其优点在于具有最小的加密粒度,具有更好的灵活性和适应性。

该软件设计中对加密粒度和算法选择的难度主要在于通过实验找到简单、可靠和容易实现的数据库加密算法,使其以最小的代价来实现系统,对数据库本身性能影响最小。

2.2 解密机制研究和实现

目前数据库应用大多基于 C/S 或 B/S 模式,因此加解密的过程可以在客户机或在服务器上完成,两种方案如图 2 所示。两种方案各有优缺点。

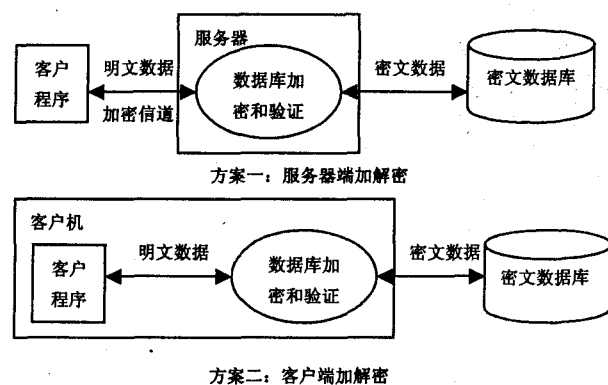


图 2 两种在数据库系统中加入数据库密码保护层的方案

方案 1 在服务器端进行操作有利于统筹规划和缓存,但是在用户数量较大时,可能会导致服务器负担过重,并且由于用户提交给服务器的数据和服务器传输给用户的数据都是明文,需要通过加密信道如 SSL 来通信,又经过了第二次加解密。

方案 2 由各客户端分别完成自身的加解密工作,有效地分散了工作量,并且由于用户提交给服务器的数据和服务器传输给用户的数据仍然是密文,因此客户端与服务器端可以直接通信而无需再加解密。当然如果需要更好的安全性,比如需要隐藏包括操作类型在内的一切信息,亦可加入加密信道。

在上述两种方案下,对于数据库加密技术使用的密钥的管理有不同的需求。当使用方案 1 时,密钥仅被服务器需要,而客户机无需使用密钥,因此密钥的管

理很简单:只需要将密钥保存在数据库服务器上,在解密过程需要时,传递密钥给应用服务器即可。但是对于方案 2,需要将密钥分布到各个客户机使得其可以完成加解密操作,将数据操作密钥存放在各客户机上不论是在层次上还是从安全角度考虑都不合适,显然必须存放在服务器上。当用户通过应用层的身份验证和系统层的存取控制时,就意味着该用户可以得到密钥。

由于客户端与服务器端之间的连接并不一定是安全连接,为了安全地传递密钥,需要有一个传输密钥来保护密钥。第一种方法,可以依靠前两层来提供传输密钥,然而并不是所有的系统都可以提供可供使用的传输密钥,并且在数据库密码保护层无法了解到应用层或系统层的密码管理体制,所以这种方法并不可行,因此不能依赖于其他层来提供传输密钥;第二种方法是在数据库密码保护层建立传输密钥,一种方案是在数据库密码保护层加入独立的身份验证或将应用层的身份验证功能移动到数据库密码保护层,该身份验证实现密钥交换适应传输密钥的需要。

2.3 数据库验证机制研究

实现验证的基本技术是使用校验和或 Hash 码来为数据生成验证码,由于 Hash 算法的冲突抵抗能力更强,因此一般宜选用 Hash 算法来为数据生成验证码。由于数据库具有文件(表)、记录、字段多个层次的概念,因此对数据库的验证同样也可以分别选用文件(表)、记录、不同记录不同字段来生成验证码的技术^[4]。

3 数据库加密与解密技术的建立

3.1 生成子密钥的字段加密技术

根据数据库数据组织的特点,一张数据表是由 M 条记录组成的,而每一条记录又由 N 个字段组成,这样一张表共有 $M \times N$ 个数据。记录 i 中字段 j 的明文可以这样表示: D_{ij} , 其中 $0 \leq i \leq M, 0 \leq j \leq N$; 记录 i 中字段 j 的密文表示为: $X_{ij} = E(K, D_{ij})$, 其中 $0 \leq i \leq M, 0 \leq j \leq N$; E 为某种对称密码算法, K 为该算法所使用密钥^[5]。其中的关键在于 K , 如果使用同样的 K 对不同的 D_{ij} 做加密,则显然对密文搜索攻击是不安全的;而使用不同的 K 对不同的 D_{ij} 来做加密则密钥数量又太大。

可以注意到,每个 D_{ij} 本身对应有两个在整张数据表中特殊的值: R_i, C_j 。其中 R_i 表示 D_{ij} 所在行的记录代码,这种特殊标记可以数据库中每一条记录的标记为 Unique 且不需要保护的字段中的数据,或者更一般的标记为 Identity(Microsoft SQL Server) 或 Auto-

Increment(MySQL) 字段中的数据^[6]。而 C_j 表示 D_{ij} 所在列的字段代码,其中最简单的办法是取字段名。这样就可以利用这两个特殊值为每一个 D_{ij} 生成一个特殊的 K_{ij} ,而且这个 K_{ij} 无需保存,可以在运行时生成。

由于 K_{ij} 需要保密,因此需要一个密钥 K 和 R_i, C_j 一起生成这个 K_{ij} ,其中 K 是严格保密的,而 R_i 无需保密。这样,生成 K_{ij} 的算法可表示为 $K_{ij} = F(K, R_i, C_j)$ 。由于 R_i, C_j 一般仅可在一张表内保持唯一,因此,对于不同的表应取不同的密钥 K ,否则会在不同的表之间生成相同的密钥。这样, K 一般表示为 TK , TK 是表密钥。不同的表有不同的表密钥。对于生成 K_{ij} 的算法 F 有以下安全要求: R1. F 必须是密码学单向函数。 R2. 必须使用 TK 才能得出 K_{ij} 。 R3. 对于每个不同的 D_{ij} 能够生成不同的 K_{ij} , 重复的概率应该尽量小。 R4. 生成的子密钥应该有足够的长度,并且无明显规律。 R5. 无法通过一个子密钥计算另一个子密钥。目前已有的 F 有以下十种:

- A1. $K_{ij} = E(K_j, R_i), K_j = E(TK, C_j)$;
- A2. $K_{ij} = K_j \oplus R_i, K_j = E(TK, C_j)$;
- A3. $K_{ij} = E(K_i, C_j), K_i = E(TK, R_i)$;
- A4. $K_{ij} = K_i \oplus C_j, K_i = E(TK, R_i)$;
- A5. $K_{ij} = E(TK, R_i \oplus C_j)$;
- A6. $K_{ij} = E(K_j, R_i), K_j = TK \oplus C_j$;
- A7. $K_{ij} = E(K_i, C_j), K_i = TK \oplus R_i$;
- A8. $K_{ij} = E(K'_{ij}, R_i), K'_{ij} = TK \oplus R_i \oplus C_j$; 或 $K_{ij} = E(K'_{ij}, C_j), K'_{ij} = TK \oplus R_i \oplus C_j$;
- A9. $K_{ij} = E(K'_{ij}, CON), K'_{ij} = H(TK, R_i, C_j)$; 这里 CON 是一个常数, H 是一个 Hash 函数
- A10. $K_{ij} = K_i \oplus K_j, K_i = E(TK, R_i), K_j = E(TK, C_j)$ 。

下面就这十种算法的安全性和效率做一下分析:

算法 A1 通过用密钥 TK 加密 C_j 首先生成一个字段密钥 K_j , 然后通过密钥 K_j 加密 R_i 产生子密钥 K_{ij} 。由于采用了两步加密,所以某个子密钥 K_{ij} 的泄露将不会引起字段密钥 K_j 的泄露;由于计算子密钥时需要 K_j , 所以同一记录中其他字段的子密钥不能从密钥 K_{ij} 推出。算法 A1 的缺点是计算一个记录中的每个元素的子密钥,需要进 2 次加密运算。这意味着加密或解密一个单个的数据需 3 倍的工作量。另外,对于一个特定字段 j 的多记录访问来说,字段 j 中每个元素的子密钥都需要用 K_j 进行加密计算获得,增加了额外运算。

算法 A2 与算法 A1 相似的是首先计算出字段密钥,但之后是按位异或运算快速生成子密钥。因为特定字段中每个元素的子密钥是由字段密钥快速生成的,

所以对于特定字段的多记录访问,这个方法是很有效的。存在的问题是,一旦某个子密钥 K_{ij} 泄露了,那么很容易计算出 K_j ,从而也就很容易计算出字段 j 中的所有的子密钥(假设记录代码是已知的),因此算法 A2 无法满足安全条件 R5。

算法 A3 与算法 A1 相似,只是调换了 R_i 和 C_j 的加密顺序,因此算法 A3 拥有和算法 A1 完全一样的安全性和效率。

算法 A4 与算法 A3 相似的是首先计算出记录密钥,但之后是按位异或运算代替了第二次加密运算生成子密钥。用这种方法加密或解密一整条记录的时间与基于记录的加密相同,对于单个字段的多记录访问,仍需要多次计算记录密钥来获得字段里每个元素的子密钥。与算法 A2 一样,当一个子密钥 K_{ij} 泄露时,算法 A4 不能保证其他子密钥的安全。这是因为一旦某个 K_{ij} 泄露了,很容易计算出记录密钥 K_i ,从而也就很容易计算出记录 i 中的每一个子密钥,因此算法 A4 无法满足安全条件 R5。

算法 A5 通过用密钥 K 加密 R_i 与 C_j 的按位异或的结果计算子密钥 K_{ij} 。这个方法的优点是只需计算一次即可求得一个子密钥。由于获得一个子密钥只需一次加密运算,所以比 A1 和 A3 快,但对于特定字段的多记录存取比算法 A2 要慢;对于单记录加密和解密,比算法 A4 要慢。用算法 A5,对于任意两个记录,一旦 $R_i \oplus C_j = R_p \oplus F_q$,那么 $K_{ij} = K_{pq}$,对于 D_{ij} 和 D_{pq} 生成了相同的密钥,并且一旦 K_{ij} 泄露, K_{pq} 也泄露,因此算法 A5 无法满足安全条件 R3 和 R5。

算法 A6 ~ A9 安全性和 A1 与 A3 相当,效率上由于只需要一次加密操作,因此较 A1 和 A3 高,但在总体上较 A2 和 A4 慢。A10 的安全性也与 A1 和 A3 相当,但需要两次加密,因此效率较低。生成子密钥的字段加密技术在密钥管理上相对简单,只需要保存表密钥 TK 即可,由于表的数量通常不会太大,因此该方案需要保存的密钥数据量很小。本算法生成子密钥并对字段加密的过程如图 3 所示,在图中 A' 表示 A 的密文,并依此类推。

3.2 基于字段的记录验证技术

针对本项目使用的生成子密钥的字段加密技术,要给出一种结合字段和记录验证的验证技术。它包括生成验证码、确认验证码两个阶段。在安全性上,仍然使用字段级的验证生成和确认,错误定位可以精确到字段级,同时将一条记录的验证码有顺序地关联到一

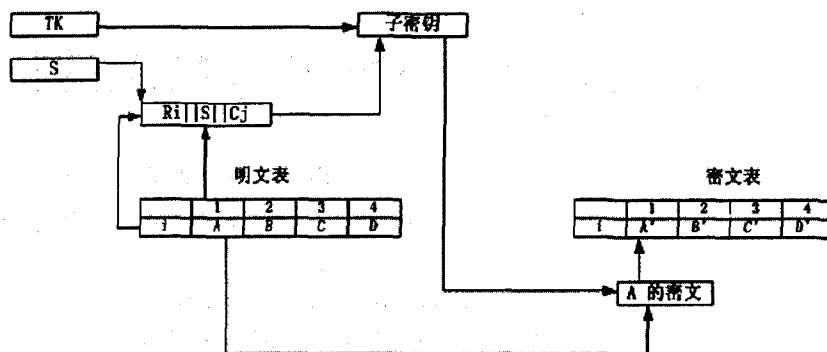


图 3 生成子密钥并对字段加密的过程

起,使得其具有基于记录的验证技术的安全性。在行上,任何并非由数据库加密与验证系统发起的行上部分列交换的动作都会导致验证无法通过;在列上,任何并非由数据库加密与验证系统发起的列上部分行交换的动作都会导致验证无法通过,除非交换整行的信息,然而此种攻击是没有意义的。因此该技术能够抵抗交换攻击。但是由于该技术只关联了记录内的信息,因此,仍然无法抵抗替换攻击。如果在属性列上也加入如上的验证码,则可以抵抗替换部分数据的攻击但无法抵抗替换整个文件(表)的攻击,并且此方法对数据库的时空效率影响过大。如何使得数据库能够抵御这种替换攻击,也是要关注的。在灵活性上,与基于字段的验证技术相同,同样可以只对需要验证的字段进行验证。

4 结束语

数据库是构建信息系统的重要基础,数据库安全也成为了信息安全的重要研究领域。目前大部分数据库的数据都是以明文存储并且没有验证机制,采用一般的身份验证与识别和存取控制技术并无法充分保障数据库的安全。立足于解决数据库中最关键的数据库验证机制等方面的理论、方法和技术问题,以关系数据库为数据模型,研究数据以密文方式存储的数据库加密与验证机制,并将基于连接的子密钥生成算法、基于用户信息的密钥管理技术和基于字段的记录验证技术融入其中,同时给出设计和实现时所涉及的 SQL 语言扩展、加密字典设计和二级密文索引设计等相关方案。为目前大多数基于关系数据库的信息系统,从源头上提供一种数据安全保障。在数据库加密方面,分析已有的数据加密技术,提出基于连接的子密钥生成算法以及基于用户信息的密钥管理技术,有效地管理生成子密钥的字段加密技术所需的密钥,适应于客户端或服务端的加解密。在数据库验证方面,提出一种基于字段的记录验证技术。适应于字段验证技术的灵

(下转第 177 页)

置声明性的事务:

```
<bean id="transactionManager" class="... DataSource-
TransactionManager">
  <property name="dataSource"><ref bean="dataSource" /
></property>
</bean>
<bean id="baseTransactionProxy" class="... Transaction-
ProxyFactoryBean" ...>
  <property name="transactionManager" ref="transaction-
Manager" />
  <property name="transactionAttributes">
    <props>
      <prop key="select *">PROPAGATION-REQUIRED
</prop>
      .....
      <prop key="select *">PROPAGATION- RE-
QUIRED,readOnly</prop>
    </props>
  </property>
</bean>
<bean id="userTargetProxy" parent="baseTransac-
tionProxy">
  <property name="target">
    <bean class="com. jaffa. app. user. domain. logic. UserIm-
pl">
      <property name="userDAO"><ref bean="userDAO" /
></property></bean>
    </property>
  </bean>
  <bean id="userFacade" class="... ProxyFactoryBean">
    <property name="proxyInterfaces">
      <value>com. jaffa. app. user. domain. UserFacade</value>
</property>
    <property name="interceptorNames"></property>
    <list>
      <idref local="userTargetProxy" /> </list>
    </bean>
    <bean id="userDAO" class="com. jaffa. app. user. dao. i-
batis. UserDAOImpl">
      <description>用户管理</description>
```

(上接第 169 页)

活性和记录验证技术的安全性。

参考文献:

- [1] 刘 念. 基于 B/S 结构的数据库加密系统的研究与实现 [D]. 成都:四川大学, 2003.
- [2] 陈 卫. 数据库加密密钥的分配和管理技术[J]. 清华大学学报:自然科学版, 1994, 34(1): 99-102.

```
<property name="sqlMapClient"><ref bean="sqlMap-
Client" /></property>
</bean>
```

Spring 的事务配置包括两个部分:其一,定义事务管理器 transactionManager,使用 DataSourceTransaction Manager 实现事务管理;其二,对各个业务接口进行定义。userTargetProxy 注入了事务管理器,此外还定义了业务接口事务管理的方法(允许通过通配符的方式进行匹配声明),有些接口方法仅对数据进行读操作,而另一些接口方法需要涉及到数据的更改。UserImpl 作为一个目标类注入事务管理器中,而其所需的 UserDAO 通过注入 UserDAOImpl 实现。

由上面介绍可以看出正如图 1 所示, Tapestry 框架通过 Spring 框架的 ApplicationContext 调用底层服务,而 Spring 框架通过将 Ibatis 的 sqlMapClient 注入 DAO 以及将 transactionmanager 注入 service,实现对底层数据库的操作。

3 结束语

介绍了 J2EE 开发中的轻量级框架组合 Tapestry + Spring + Ibatis,并通过实例说明组合框架的应用。应用 Tapestry + Spring + Ibatis 框架构建 Web 应用程序具有很好的扩展性、可维护性,并能充分发挥三者的优势,因此该组合框架具有很好的应用前景。

参考文献:

- [1] Howard M, Ship L. Tapestry in Action[M]. [s. l.]: Manning, 2004.
- [2] Apache Software Foundation. What is Tapestry[EB/OL]. 2008. <http://tapestry.apache.org/tapestry5/>.
- [3] Begin C. iBATIS Database Layer[EB/OL]. 2006. <http://www.Ibatis.com/>.
- [4] 夏 昕. Ibatis2.0 开发指南[EB/OL]. 2004. <http://www.matrix.org.cn/resource/down/581.html>.
- [5] 龚雪冰,何 彪. 基于 Tapestry + Spring + Hibernate 框架的 Web 应用[J]. 计算机技术与发展, 2007, 17(4): 131-135.

- [3] 戴一奇,尚 杰,陈 卫,等. 一种新的数据库加密密钥管理方案[J]. 清华大学学报:自然科学版, 1995, 35(4): 43-47.
- [4] 崔国华,汤学明. 数据库中加密机制的实施研究[J]. 密码与信息, 2001(2): 84-90.
- [5] 戴一奇,尚 杰,苏中民. 密文数据库的快速检索[J]. 清华大学学报:自然科学版, 1997, 37(4): 24-27.
- [6] 余祥宜,谭谦仁. 加密数据库通用快速查询算法研究[J]. 华中理工大学学报, 2002, 28(12): 27-29.