基于 B/S 结构的数据库加密研究*)

尚 晋[1] 徐江峰[2] 黄小粟[2]

(重庆电子职业技术学院计算机二系 重庆 400021)[1] (郑州大学信息工程学院 郑州 450052)[2]

摘 要 网络技术的快速发展,数据库的广泛应用,使得数据库安全技术研究已成为信息安全研究的主要课题之一。在对数据库加密技术进行研究的基础上,给出了一个基于 B/S 结构的数据库加密方案,并利用 JSP 技术实现了上述方案。理论分析和实验结果表明,该方案可以提高数据库系统的安全性能,并且实现简单。

关键词 浏览器/服务器,数据库加密,Java 安全,密钥

Research on Database Encryption Based on B/S Architecture

SHANG Jin[1] XU Jiang-Feng[2] HUANG Xiao-Su[2]

(The Second Department of Computer, Chongqing Electronic Profession College, Chongqing 400021)[1]

(School of Information and Engineering, Zhengzhou University, Zhengzhou 450052)[2]

Abstract With the development of network technology and database application, database security has been one of problem of information security. On the research of database encryption technology, a database encrypted mode based on B/S architecture is introduced. Theoretical analysis and experimental results demonstrate that the database encryption system designed by JSP can improve the security of the database system.

Keywords Browse/Server, Database encryption, Java security, Key

1 引言

随着网络技术的不断发展,计算机在社会生活的许多方面都得到了应用。然而在计算机给人们的工作和生活带来方便的同时,各种安全隐患也应运而生了。为了提高网络系统的安全性能,人们采用了许多保护手段,如:在系统中增加防火墙和入侵监测系统等,但是这些措施仍然不能避免系统遭受破坏,不能完全保证系统的安全性。数据库系统作为信息系统的核心部件,数据库文件作为信息的聚集体,显然其安全性将是信息系统安全性的重要指标。使用没有安全保证的数据库,一个信息系统不可能具有高的安全性能。所以,数据库的安全问题已成为信息安全领域的主要研究课题之一。

为了提高数据库的安全性能,数据库厂商也提供了许多安全措施,如用户分类、数据分类、审计功能、数据库备份与恢复等。然而这些功能显然是不够的,其主要原因是数据库中数据是以明码文方式存放的,只要黑客避开了系统认证,进入到数据库系统中,他就可以盗取或修改数据库中他需要的任何数据。此外,威胁数据库安全的并不完全是外部人员,系统内部的管理人员或企业内部职员对数据库安全的威胁更大,此时防火墙、入侵检测等防护措施没有任何作用,他们很容易实现对数据库的拷贝、修改等操作,从而给企业造成重大损失。因此,为了进一步提高数据库的安全性能,对数据库实行加密存储将是非常必要的。

随着 Internet 的快速发展,B/S(Browser/Server)结构已经替代 C/S (Client/Server)结构成为网络信息系统中主要的服务方式。客户端只需装上操作系统、网络协议软件、浏览器,就可以实现信息的浏览和数据的交换,不受平台的约束,所有的操作基本上都在服务器端进行。然而由于 Internet 是一个开放的系统,其本身有很大的安全隐患,只要 Web 站点和 Internet 连通,任何人都可以对服务器进行访问,从而使非法用户有可能使用秘密渠道直接访问数据库,非法盗取或修改数据库数据,对数据库造成破坏。如果数据库数据进行加密存储,非法用户即使获得了数据库数据,他也无法使用,从而提高数据库的安全性。

本文在对数据库加密技术进行深入研究的基础上,提出了一个基于 B/S 结构的数据库加密方案,并利用 JSP 技术实现了上述方案。对该方案的安全性分析表明,该方案可以提高 B/S 结构后台数据库的安全性能。

2 数据库加密的设计要求

数据库系统能够快捷和有效地完成各种数据操作是网络信息系统正确运行的前提。因此,对数据库进行加密处理应满足如下基本要求:

(1)加密后的数据仍然存放在数据库系统中;

(2)加密后的数据库,其操作仍然是可行和有效地;

(3)合法用户对数据库的访问,效率不受明显影响;

(4)严格的身份认证,使非法用户不能对加密数据进行正确访问。

为满足上述要求,在对数据库进行加密时,设计者需考虑下面几个主要问题。

2.1 数据加密层次

对数据库进行加密,可选择的层次主要有:操作系统层、DBMS 内核层和 DBMS 外围层。在操作系统层实现加密,由于无法辨认数据库文件中的数据关系,因而无法根据数据关系进行加密,只能对数据库文件进行文件加密,这对于大型数据库来说没有实际意义。在 DBMS 内核层实现加密,数据的加密和解密工作可以在物理存取之前完成,从而不影响数据库的各种操作,并且加密效率高,但需要有 DBMS 开发商的支持,实现非常困难。在 DBMS 的外围层实现加密,是将数据库加密系统做成 DBMS 的一个外层工具,在实现时既可以

*)本课题得到国家自然科学基金(60074034)的支持。尚 晋 硕士研究生,讲师,研究领域:系统集成和信息安全。徐江峰 博士,副教授,研究领域:信息安全及混沌加密通信。黄小粟 硕士,研究领域:信息安全及数据库加密。

充分考虑数据库中的各种数据关系，又不需要开发商的支持，是一种切实可行的加密方法。

2.2 加密粒度

加密粒度是指数据库加密的最小单位。数据库中的数据根据层次可以分为数据表、数据记录、字段和数据项，所以数据库数据加密的加密粒度通常有文件级、字段级、记录级和数据项级。加密单位越小，灵活性越强，适用范围越广，但实现难度就越大。在实际应用时，要根据不同的安全需求，选择不同的加密粒度，实现既保证数据安全又便于操作的目标。

2.3 加密算法

加密算法是一些公式和法则，实现把明文转换成密文；解密算法是加密算法的逆运算，实现把密文转换成明文。而密钥则是加密算法对明文进行加解密处理的钥匙。根据加密密钥及解密密钥是否相同，加密算法分为对称加密与非对称加密两大类，常用的对称加密算法有 DES、AES 等，非对称加密算法有 RSA、椭圆加密算法等。与一般的数据文件不同，数据库中的数据有其自身的显著特点：数据量大、大量数据之间存在确定的逻辑关系。所以，数据库加密算法的选择和设计应满足如下要求：

(1)数据库加密以后，数据量不应明显增加；

(2)攻击者仅依据数据间的逻辑关系破解数据库中的加密数据将是十分困难的；

(3)某一数据加密后，其数据长度不变；

(4)加/解密速度要足够快，数据操作响应时间应该让用户能够接受。

2.4 数据库加密的限制

数据库中的数据表与数据表之间、字段与字段之间常常存在着密切的联系；为了达到迅速查询的目的，数据库文件中建立有索引；数据库的许多操作都带有条件选择，而条件中的选择项必须是明文。因此，在数据库加密时，要充分考虑这些限制条件，否则将会造成数据库中的数据失去统一性，索引失去作用，数据库操作失败或给出错误的操作结果。

3 基于 B/S 结构的数据库加密设计和实现

3.1 系统结构设计

基于 B/S 结构的网络系统，其应用程序都在服务器端，客户端只需要 Web 浏览器，不需开发专用的应用程序，用户通过浏览器实现与服务器的信息传递和访问。因此数据加密在 Web 服务器与后台数据库之间实现，加密系统结构如图 1 所示。

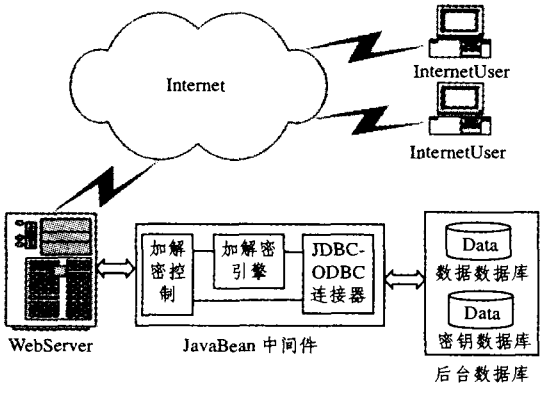

图 1 基于 B/S 结构的数据库加密结构

从图中可以看出，终端用户通过 Web 浏览器向服务器发出 HTML 请求，Web 服务器对需要进行数据库访问的数据交加密/解密中间件处理。当数据需要加密存储时，就由加密组建完成加密，并把加密后的数据提交数据库存放；当需要访问加密数据时，就由解密组件首先完成解密，而后再交给 Web 服务器；否则，Web 服务器就通过 JDBC-ODBC 连接器直接与后台数据库连接。当然，在数据的加密和解密过程中，必须要有密钥才能实现，所以加密/解密组件中包含有密钥管理模块。

3.2 系统开发平台

网络的广泛应用带来了网络应用程序开发工具的快速发展，常用的网络开发工具有 CGI、ASP 和 JSP。JSP 以 Java 为脚本语言，具有平台无关性和扩展能力强的特点，对数据库广泛支持，并且它与 Internet 紧密结合，克服了原来 CGI 编程中存在的效率低下，实时响应要求不高，系统资源占用率高等缺点。另外，Java 还具有健壮的、功能强的安全体系结构，其安全框架由三个 API 组成：Java 认证和授权服务(JAAS)、Java 安全套接字扩展(JSSE)和 Java 加密扩展(JCE)。JAAS 提供了一种灵活的、说明性的机制，用于对用户进行认证并验证他们访问安全资源的能力；JSSE 定义了通过安全套接字层(SSL)实现安全 Web 通信的 Java 机制；Java 加密扩展(JCE)是 Sun 的加密服务软件，包含了加密和密匙生成功能，它没有规定具体的加密算法，但在其软件包中包含有许多加密算法。JDBC 是由 JavaSoft 公司开发的 Java 语言访问数据库接口，它能够保证 Java Servlet 和 Web 数据库之间高效、安全的连接。JSP 的上述技术特性，使得它可以很好地用来开发基于 B/S 结构的数据库加密系统。

3.3 加密系统设计

根据前面的分析，在一个数据库加密系统中，应该包含身份认证、加密算法和密钥管理等功能。各模块的功能及实现方法描述如下：

1. 用户验证模块　该模块通过 Interne 由 Web 服务器直接与客户进行对话。其功能是对用户身份进行验证，防止未授权人员使用系统。如果用户注册成功，则从密钥数据库中获得该用户密钥，并确定需要加密处理的数据。

2. 加密算法选择　加密算法是数据加密的核心，一个好的加密算法产生的密文应该频率平衡，随机无重码规律，周期很长而又不可能产生重复现象，窃密者很难通过对密文频率、重码等特征的分析获得成功。除了上述要求，数据库加/解密算法还必须满足数据库系统的特性，算法应该可以快速处理数据库中的量大、关系紧密的数据。因此，在我们的加密实现中，我们选择了加解密速度较快的 DES 算法。

3. 加密粒度选择　数据库加密的粒度，就是加密的最小数据单位。从前面的分析知道，数据库加密的粒度有文件、记录、字段和数据项。文件级加密实现简单，但严重影响数据库操作的效率；数据项加密安全性可能是最高的，但效率低、密钥管理极为困难；记录加密实现的是对一个完整的实体进行加密，操作相对简单，然而解密时需要对一个记录中的所有字段进行，无法实现对一个记录中特定字段解密。因此，本系统选择字段作为加密粒度，它虽然也有不足，但加密/解密时具有比较好的灵活性和适应性。然而，在实现字段加密时，需要解决的一个关键问题是密钥管理。

4. 密钥管理模块　密钥管理是对数据密钥和用户密钥的生成和存储进行管理。当产生一个新的数据库用户时，系统采用投币法产生 64 位二进制数据作为用户密钥，并用户口令加密后放在密钥数据库中；当建立一个新的数据表时，对每个需加密的字段系统自动产生一个数据密钥，以后存入该字

段的数据就用该数据密钥进行加密，并且该密钥由用户密钥加密后也存入密钥数据库中。因此，要想对加密字段进行解密，需要有数据密钥，而数据密钥的获得，必需有户密钥才行，用户密钥的获得，只有输入正确的用户口令才可以。从中可以看出，密钥的管理是多层次的，没有密钥或仅知道部分密钥，是不可能对数据库加密数据进行破解的，这就大大提高了数据库数据的安全性能。

5. *数据库连接模块* 数据库连接由几个 JavaBean 构成，实现 Web 服务器与后台数据库的连接，并读取和存入数据。本系统通过 JDBC-ODBC 与数据库建立连接，并将用户数据请求转换为相应的 SQL 语句，对数据库进行查询、添加、删除和修改等操作。

6. *加密/解密引擎* 数据库加/脱密引擎是数据库加密系统的核心部件，负责在后台完成数据库信息的加/解密处理，对应用开发人员和操作人员是透明的。同数据库连接一样，该模块也是做成 JavaBean 的形式，当加解密控制确定用户数据不需要加密/解密处理的时，由连接模块和数据库直接连接；当需要对数据加密时，由 JCE 产生密钥，并对数据进行加密；对于需要解密的数据，则首先从密钥库中分别取出用户密钥和数据密钥，并利用用户密钥解密数据密钥，而后再利用数据密钥对加密数据进行解密。

3.4 系统安全性分析

数据库中敏感数据经加密系统处理后，变成形式上无规则的乱码，这样即使非法使用者窃取了数据库文件，他仍然难以得到所需的信息。另外，在本系统中由于密钥管理采用的是多级加密机制，无论是数据密钥还是用户密钥都是加密形式存储，因此即使非法用户打开了数据库，由于他不知道用户口令，无法解密用户密钥，更不可能得到数据密钥，因此他无法对加密数据进行解密，这就大大提高了关键数据的安全性。

结束语 数据库加密方面的探索工作已进行多年，随着网络技术的发展，基于数据库的安全研究出现了新课题，就是如何在网络的环境下提供一个安全的数据库环境。在实际应用中要实现一个有效、快捷和真正安全的网数据库是十分困难和复杂的。本系统对基于 B/S 结构的数据库加密技术进行了研究，设计了一个加密系统，并利用 JSP 技术进行了实现。理论分析和实验结果表明，该系统可以提高数据库系统的安全性，有效抵御非法窃取和访问数据库数据。但在网络环境下，数据在网络上传输时仍是明文数据，仍然存在被非法窃取的可能性。解决这方面的问题要运用 JAVA 的数字化身份认证体系和数据的安全传输等一系列的安全措施来构建起一个安全体系。这也是我们下一步要做的研究工作。

参 考 文 献

1 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
2 Stallings W. 密码编码学与网络安全—原理与实践(第 2 版)[M]. 北京：电子工业出版社，2001
3 Garms J. Java 安全性编程指南[M]. 北京：电子工业出版社
4 李广鑫，马志欣，等. 基于 B/ S 结构的远程实时监测系统[J]. 计算机应用研究，2003，10：147～150
5 张华桁，宋立群，柯科峰. B/S 构架信息系统的安全策略研究与开发[J]. 计算机工程与应用，2004，13：159～162
6 朱鲁华，陈荣良. 数据库加密系统的设计与实现[J]. 计算机工程，2002，28(8)：61～63

(上接第 69 页)

文件相关的客户端键值区间的中点和下载速度记录，利用 K-means 聚簇算法处理这些键值区间中点和下载速度数据，可以找出最需要副本的键值点，根据这些键值点，我们可以找到一些站点，在每个这样的站点上做一个文件副本。

如果文件的使用频率下降，需要的副本数可能需要减少。根据公式(8)，如果系统需要减少文件的副本数时，根据需要可以将使用频率最小的几个副本撤销。

3.4 副本的重置

对于一个副本节点 S_{bak}，它上边的文件副本除了会被上面的过程给强迫释放外，另一种情况就是主动转移或释放。当 S_{bak} 上需要创建新的文件副本而存储空间不足时，这种情况就发生了，此时站点 S_{bak} 就会试图把站点 S_{bak} 上所有副本中使用频率最低的副本 $F1$ 给转移到它在 *GFS-Net* 中的邻居节点上。邻居节点的选择的次序的依据是副本节点 S_{bak} 的键值区间中点到邻居的距离。如果 $F1$ 的使用频率都不比邻居节点 Sn 上的任何一个副本的使用频率高且 Sn 上没有足够的空间用来放置副本 $F1$，则找另外一个邻居节点。如果副本 $F1$ 的使用频率比全部的邻居站点任何一个副本的使用频率低且邻居都没有足够的空间用来放置副本 $F1$，则 S_{bak} 把 $F1$ 直接释放。相反，如果找到一个邻居节点 Sn：如果 Sn 上有足够的空间来盛放副本 $F1$，S_{bak} 就会把 $F1$ 转移给 Sn；如果副本 $F1$ 的使用频率比 Sn 上的某个文件副本 $F2$ 的使用频率高，则尝试用类似 $F1$ 的方法将 $F2$ 移出，然后再查看 Sn 的空间是否够用，如果够用则将 $F1$ 放到站点 Sn，否则继续分析是否有副本可以移出。重复上面的操作直至找到一个有足够的空间的邻居节点放置副本 $F1$。如果副本节点 S_{bak} 上的存储空间仍不满足要求，则继续上述过程直到 S_{bak} 上的存储空间满足要求为止。节点 S_{bak} 上负载太重也可以用上述方法处理其上的副本以降低站点的负载。

该动态复制算法，使用星形拓扑结构，利用近期用户请求的频率和分布预期将来的用户请求的频率和分布，动态调整副本的数目和分布。在用户的访问模式带有一定的地域集中性情形下，与级联复制相比除了具有更高的复制灵活性之外还具有更好的性能。

总结 本文的主要贡献：1)完善了 GFS-Net 的搜索算法；2)提出了适用于 GFS-Net 的动态复制算法。该动态复制算法包括两个部分：依据副本的请求频率增减副本数的方法；根据下载响应时间较长的客户端的分布(利用客户端上的覆盖区间的中值)布置新增副本位置的方法。这些方法平衡了各个站点的下载冲击，降低用户文件下载的响应时间。

对于大文件，由于客户端请求文件时返回的是所有副本的地址信息列表，客户端可以选择多个源端来进行条带传输(stripped transfer)以提高下载速度，而如何选择源端和如何协调各个下载线程是我们下一步的工作内容。另外，公式(8)中常数 a，b 和 K 的选择虽然通过在远程教育平台的应用实验过，但还要通过其他应用来总结出合理的取值。

参 考 文 献

1 李庆虎. 基于 P2P 架构的网格文件系统研究：[博士论文]. 清华大学，2004
2 Li Q H, Wang J M, Lam K Y, Sun J G. Gridfs: A web-based data grid for the distributed sharing of educational resource files. Advances in Web-based Learning Intl. Conf. ICWL 2003, Lecture Notes in Computer Science 2783, 2003. 81～92
3 Li Q H, Wang J M, Sun J G. GFS-Btree: A scalable peer-to-peer overlay network for lookup service. The Second International Workshop on Grid and Cooperative Computing, LNCS3032: 340～347
4 Ranganathan K, Foster I. Identifying dynamic replication strategies for a high performance data grid. In: Proc. of the Intl. Workshop on Grid Computing, Denver, Colorado, Nov. 2001
5 Gwertzman J S, Seltzer M. The case for geographical push-caching. In: Proc. Fifth Workshop on Hot Topics in Operating Systems, May 1995
6 Michel S, Nguyen K, Rosenstein A, Zhang L, Floyd L, Jacobson V. Adaptive web caching: Towards a new global caching architecture. In: Proc. of the 3rd Intl. WWW Caching Workshop, 1998