基于 AES 与 HASH 的软件数据保护①

吕君可

(浙江师范大学 行知学院，金华 321004)

摘 要：从数据存储与用户身份识别两方面提出了软件数据保护的方案。利用对称加密算法 AES 实现数据库敏感信息的加密存储；利用 SHA512 结合注册表实现软件的身份识别，并在 VB2008 中给出了实现过程，从而实现软件数据的双重保护。

关键词：AES；HASH；数据存储；身份识别；SHA512

Protection of Software Data Based on AES and HASH

LV Jun-Ke

(Xingzhi College, Zhejiang Normal University, Jinhua 321004, China)

Abstract: From the two aspects of data storage and user identification, this paper provides a program to protect the software data. It uses symmetrical encryption AES to achieve the encryption storage of the sensitive information in the database; uses SHA512 and regedit to achieve the software identification, and supplies the achieving process in VB2008, thus the software data can be double protected.

Keywords: AES; HASH; data storage; identification; SHA512

1 引言

随着信息化社会的发展，越来越多的个人信息软件使用了数据库技术，大量的个人信息被存储到数据库中，其中不乏敏感信息，因此，数据库的安全性以及对敏感数据的防窃取和防篡改的保护至关重要。目前大部分数据库都是以明文存储，一旦有非法入侵者进入数据库，就极易造成重要数据的泄露，例如数据库管理员就可以毫无阻碍地访问其中的重要数据[1]。因此，除了在外部加强数据库的安全性以外[2]，更有效的方式就是对数据库内的数据进行加密储存；而软件的身份识别是保护软件数据的另一重要关口，否则一旦获取合法用户的身份，从软件层面就可以窃取用户的敏感信息。

本文采用高强度、高版本的加密算法，结合注册表操作，从数据存储与软件用户身份验证两方面提出了软件数据的双重保护方案，增强了破解难度，可以有效地保护软件的数据信息。所涉及的程序都在 Winxp 操作系统，VS2008 软件设计平台中调试通过。

2 数据加密存储

软件中的敏感信息不能直接存储，否则一旦数据库被非法获取，里面的信息将毫无秘密可言。DES 是最常用的对称加密算法，由于技术的发展，逐渐暴露出密钥相对过短的弊端，已于 1997 年被成功破解。

2.1 AES 算法描述

AES 是继 DES 之后新一代的数据加密标准。1997 年，美国国家标准技术研究所(NIST)发布公告征集新的加密标准，2000 年 10 月 2 日，NIST 正式宣布由比利时 Joan Daemen 和 Vicent Rijmen 设计的 Rijndael 算法被不加修改地作为 AES。Rijndael 是具有可变分组长度和可变密钥长度的分组密码，其分组长度和密钥长度均可独立地设定为 32 比特的任意倍数，最小值为 128 比特，最大值为 256 比特[3]。该算法经验证是所有候选算法中安全性能最高、运行速度最快，又是迭代分组的密码算法，因此在使用上更加灵活、安全。

① 收稿时间:2010-07-09;收到修改稿时间:2010-08-22

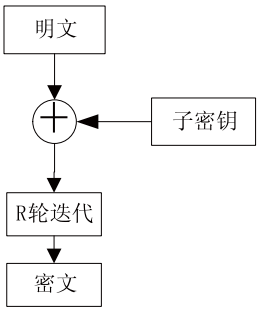

图 1 AES 算法框图

2.2 AES 算法.Net 实现

按加密的粒度，对数据库中的数据可以进行记录加密，也可以进行字段加密，还可以对数据元素进行加密。以下是在 Visual Studio 2008 软件设计平台中 vb.net 实现 AES 数据加密与解密的过程，以数据元素加密为例。

首先引入命名空间：

```
Imports System.ComponentModel
Imports System.Security.Cryptography
Imports System.Text
Imports System.IO
```

变量定义：

```
Dim encoding As New UTF8Encoding
Dim AESAlg As New AesManaged
Dim HashAlg As HashAlgorithm = New SHA256Managed
```

数据加密存储加密实现：

```
' 变量 data 为需要加密的明文数据，key 为密钥
' 检查密钥长度
If Not key.Length = 32 Then
    key = HashAlg.ComputeHash(key)
End If
Dim IV As Byte() = GenerateIV(key)
' 创建加密对象
Dim Eptor As ICryptoTransform = AESAlg.CreateEncryptor (key, IV)
' 创建数据加密内存对象
Using mStream As New MemoryStream()
  Using cStream As New CryptoStream (mStream, Eptor, CryptoStreamMode.Write)
    ' 将数据写入加密流
    cStream.Write(data, 0, data.Length)
  End Using
  ' 返回加密后的结果
  Return Convert.ToBase64String (mStream.ToArray)
End Using
```

数据加密存储后，如需访问，应该先解密数据然后再返回解密后的数据。

```
' 解密过程
Dim txtBytes As Byte() = Convert.FromBase64String(encoding.GetString(data))
' 解密数据
Using mStream As New MemoryStream(txtBytes)
  Dim decryptedData(txtBytes.Length) As Byte
  Using cStream As New CryptoStream(mStream, AESAlg.CreateDecryptor(key, IV), CryptoStreamMode.Read)
    ' 将数据读入解密流中
    cStream.Read(decryptedData, 0, decryptedData.Length)
  End Using
  ' 返回解密结果
  Return encoding.GetString(decryptedData)
End Using
```

以上实现过程给出的是字节类型的参数，如果是字符类型，则还需进行转换。转换语句为：

```
Dim dataBytes As Byte() = encoding.GetBytes(data)
```

程序的运行结果如图 2 所示，密钥根据初始数据生成。在实际使用过程中，密钥可结合自己程序的需求来设置，比如与数据字段相结合，或者与系统硬件信息结合，如硬盘序列号、主板序列号等。

图 2 加密/解密效果图

3 软件用户身份保护

用户在获取软件使用权之后，身份识别是软件的第一道防护墙。目前，很多软件将用户登录的密码也保存在数据库中，如果非法用户获取了软件中涉及的数据库，通过种种手段就能将登录密码破解，从而可获得合法用户的使用权限，从使用者层面窃取相关资料。因此，合法用户身份信息的保护也至关重要。

基于上述考虑，用户的身份信息不能直接保存到数据库中。为了保证密码的安全，除了考虑密码的存放方式之外，另一方面就是对密码本身加密，得到一个加密的密码，将密码存入注册表中，通过对注册表的操作，可以将密码随便放到注册表中，要想找到密码字符可谓大海捞针。

3.1 HASH 算法

Hash 算法是一种将任意长度的消息压缩到某一固定长度的消息摘要的函数，利用它可以对数据产生一个短的“指纹”(即消息摘要)。它是一种单向算法，一旦数据被转换，将无法再获得其原始值[4]。Hash 算法种类很多，本文采用 SHA512 算法(SHA 系列算法中的最高版本)，将明文密码经过 SHA512 运算后的摘要存储到注册表中，身份识别时根据用户输入的密码生成摘要与保存的摘要对比，从而验证信息是否正确。

3.2 获取密码 Hash 值

在 Visual Studio 2008 开发环境中采用 SHA 获取密码 Hash 值，同样需要引入命名空间，方法同上。创建函数 getshahash，参数 input 也就是明文密码：

```
Public Function getshahash(ByVal input As String) As String
    ' 创建新的 SHA512Managed 对象
    Dim shahasher As New SHA512Managed
    Dim data As Byte()
    ' 将输入字符串 input (明文密码)转换为数组并计算摘要
    data=shahasher.ComputeHash(Encoding. Default. GetBytes(input))
    Dim sbd As New StringBuilder()
    Dim i As Integer
    ' 将摘要数据转化为字符串
    For i = 0 To data.Length - 1
        sbd.Append(data(i).ToString("x2"))
    Next
    ' 返回结果
    Return sbd.ToString
End Function
```

该算法使用 SHA512Managed 对象来获得密码摘要，通过 SHA512Managed 对象的 ComputeHash 方法计算数据的 Hash 值。如果要采用其他的 Hash 算法，只需更改创建对象语句即可。

3.3 存储密码摘要

在获得了密码摘要后，本文以将密码 Hash 值保存到注册表中的 hkey_current_user\software\users 下的 password 中为例，介绍实现过程。设计存储函数 savepsw 如下：

```
    ' 参数 psw 对应函数 getshahash 中的 input
    Dim key As RegistryKey
    Dim mykey As RegistryKey
    ' 获得 hkey_current_user\software 子键，设置 writable 为 true
    key = My.Computer.Registry.CurrentUser.OpenSubKey("software", True)
    mykey = key.OpenSubKey("users", True)
    If (mykey Is Nothing) Then
    ' 创建子键并返回创建的子键对象
        mykey = key.CreateSubKey("users", RegistryKeyPermissionCheck.ReadWriteSubTree)
    End If
    ' 创建值并设置值的内容，将明文密码经过 Hash 运算后的摘要结果存入注册表中
    mykey.SetValue("password", getshahash(psw), RegistryValueKind.String)
    mykey.SetValue("userpassword", "true", RegistryValueKind.String)
    ' 上述语句中 getshahash(psw)的作用是调用函数
```

getshahash 获取参数 psw 的密码摘要。

要验证身份是否正确，需要获得被判断密码的 Hash 值，然后与保存的密码摘要进行比较，如果完全相同，则通过身份验证。这样设计的目的是，即使密码被人非法窃取，只要不是在本机上使用软件，他就没办法获取注册表中的信息，从而保障合法用户的权限。身份验证流程如图 3 所示：

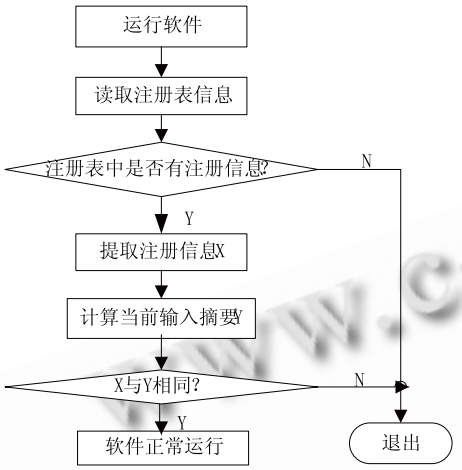

图 3 身份验证流程图

获取注册信息的关键语句：

首先获得 hkey_current_user\software 子键：

key = My.Computer.Registry.CurrentUser.OpenSubKey("software", False)

mykey = key.OpenSubKey("users", False)

然后读取注册表中的摘要信息：

Pswhash=mykey.GetValue("password").ToString

4 结束语

本文分析了软件中敏感信息的安全性问题，综合采用数据加密、身份认证等安全机制，给出了 VS2008 环境中的实现过程。所采用的算法加密强度高，破解难度大，可以有效地保护软件的数据信息。

参考文献

1 朱鲁华,陈荣良.数据库加密系统的设计与实现.计算机工程,2002,28(8):61－63.
2 李彦秋.面向数据库外层的安全技术研究[硕士学位论文].东营:中国石油大学,2009.
3 Daemen J, Rijmen V.谷大武,徐胜波译.高级加密标准(AES)算法—Rijndael 的设计.北京:清华大学出版社,2003.31.
4 段钢.加密与解密.北京:电子工业出版社,2008.131－139.

(上接第 177 页)

的集群软件 RAC 进行集群配置。软件的配置布局为，在图 2 的 B1 上安装 1 个数据库和 cluster ware 集群软件，在 B2 和 B3 上分别安装 1 个数据库。具体配置步骤如下：

1) 集群环境准备

• 配置文件修改。包括/etc/hosts、 /etc/sysctl.conf 等文件

• 创建用户/用户组和安装目录

• 配置 SSH，修改 oracle 用户配置文件

• 为共享磁盘分区

• 安装配置 OCFS2 和 ASM

2) 安装集群软件

3) 安装数据库软件

4) 创建数据库，并配置网络

5) 集群启动/停止

经过以上三步，一个完整的基于虚拟化的服务器集群实例配置完成。

5 结束语

在 Linux 平台之上，利用虚拟化技术，结合负载均衡调度算法以及集群技术，可以大幅提高企业现有服务器的利用率，从而保证企业业务的持续性和数据的完整性，节省企业大量的 IT 成本。鉴于该方案带来的优势，相信 Linux 服务器的虚拟化与集群技术在企业服务器这一广阔的领域里会得到广泛的应用。

参考文献

1 王建红.浅析 Linux 虚拟化技术.湖北示范学院学报(自然科学版),2008,28(1):62－64.
2 庞辽军,王力,李慧贤.基于集群技术的 Linux 虚拟服务器.计算机工程与应用,2003,39(14):161－163.
3 张静.RAID 技术工作原理分析及实现.科技信息(学术版),2008,(32):329－330.
4 何禹,胡宇鸿,王一波.虚拟化技术在校园网数据中心的应用.电子科技大学学报,2007,36(6):1461－1464.
5 林纲,张治辉.Linux 内核地址映射机制分析及实现.计算机与数字工程,2005,33(7):118－120.
6 陈品华.虚拟化技术在中小企业的应用.微计算机信息,2010,3(5):220－221.