数字签名在OpenType字库中的分析研究

赵 青,唐英敏
(北京大学 计算机科学与技术研究所,北京 100085)

摘 要:介绍了数字签名的方法和过程,分析了OpenType矢量字库的表结构和与数字签名相关的表的内容。结合数字证书技术,应用于OpenType矢量字库上,将数字签名嵌入字库表中。给出了基于Windows平台的字库数字签名的具体实现步骤和结果的验证过程,完成数字签名的计算和写入。将此技术应用于字库生产过程中,实现了字库的完整性、安全性、真实性和可信性,收到了良好的效果。

关键词:矢量字库;数字签名;OpenType
中图分类号:TP391 **文献标识码**:A **文章编号**:1673-629X(2008)08-0020-03

Analysis and Research of Digital Signature in OpenType Font

ZHAO Qing, TANG Ying-min
(Institute of Computer Science and Technology of Peking University, Beijing 100085, China)

Abstract: The theory and process of digital signature is introduced in this paper and both the structures and contents of the font tables relevant to it are analyzed. Combined with the digital certificate technique, the digital signature is implemented in the OpenType vector font by inserted in font tables. The concrete approach to the calculation of digital signature and the validation process of result is put forward based on the Windows platform. Applied to font production, digital signature technique accomplishes the integrality, security, authenticity and creditability of font and achieves good effect.

Key words: vector font; digital signature; OpenType

0 引 言

随着Internet在全球范围内的迅速发展和普及,世界各地的用户都可以快捷方便地下载自己需要的软件。由此而产生的安全问题日益突出。首先,软件内部可能嵌入恶意代码,窃取用户数据或者暗中操控宿主计算机从事非法操作;其次,软件中存在的潜在错误可能会导致用户数据丢失或者操作系统崩溃。第三,软件可能伪装成其他类型软件,骗取用户信任。

同所有其他软件一样,字库软件也面临着这些安全问题。解决这些问题的方法之一就是使用数字签名。数字签名相当于给软件一个身份证明,用户可以查看这个身份证明,再决定是否下载安装。为了防止签名伪造,引入数字证书,结合不可逆的哈希算法和公钥密码体制,从技术上保障签名的安全性、可靠性和真实性。

OpenType是Adobe公司和Microsoft公司于2000年合作制订推出的新字形描述格式。这种格式保存了汉字的曲线轮廓信息,具有占用空间小、显示效果优良和大小随意缩放的特点,在Windows系统字库中占有重要的地位。OpenType矢量汉字字库一直在我国各行各业中得到广泛的使用,如印刷、机械、广告、装饰等,其多种类的汉字字体满足了各个行业的不同需求。由于字库的下载和安装非常简单,而且是通过输入法软件或者排版软件间接使用,这种隐蔽性很难保证字库的质量和安全性。数字签名为字库提供了可靠的身份证明,可以确认字库的身份、标识字库生产商,防止假冒和篡改,保证用户的使用安全。

收稿日期:2007-11-30

作者简介:赵 青(1982-),女,山东青岛人,硕士研究生,研究方向为图形与文字信息处理;唐英敏,博士,高级工程师,研究方向为图形与文字信息处理。

1 数字签名技术

1.1 基本原理

数字签名[1]是一种保证数据真实性的数字技术,它是为了防止数据被篡改、保证安全性的有效手段之一。数字签名技术可以建立在公钥密码体制[2]之上。所谓公钥密码体制是指每个实体都有一对互相匹配的密钥:公开密钥(公钥)和秘密密钥(私钥)。实体通过私钥进行加密和签名;其他用户则可以通过公钥进行

数据的解密和验证。

首先,选择恰当的鉴别函数(H),这个鉴别函数通常是一个单向哈希算法。使用鉴别函数对整个文件进行哈希运算,得到一个固定长度的摘要作为鉴别标识,将此鉴别标识用私钥(KR)按照某个公钥加密算法进行加密,得到的运算结果即为数字签名的内容。由于哈希算法是不可逆的,私钥足够长且不被其他人所知,所以数字签名很难伪造,也就保证了签名的真实性。

1.2 签名的验证过程

操作系统将得到的签名文件使用相同的哈希算法构造出一个新的摘要,然后使用公钥(KU)解密数字签名的内容,将两个摘要进行比较,如果相同,则说明此软件没有被篡改过,并且确实是证书所声明的厂商发布的,而非伪造,这样就保证了软件的完整性、可靠性和安全性。

1.3 数字证书

在实际商业应用中,上述所使用的一对密钥常常是包含在数字证书[3]中。数字证书是由权威认证机构(CA)颁发给单位或者个人的电子数据文档。它使用公钥密码体制,标明了拥有者的身份,具有唯一性和不可伪造性。一般情况下,数字证书包含两个单元:一个是证书文件(*.cer)和一个密钥文件(*.pvk)。证书文件存储了拥有者的信息、公开密钥和CA的签名,可以对外公开或者直接嵌入到应用程序中;密钥文件存储了秘密密钥,必须保持绝密。数字签名过程使用证书文件,签名的验证过程使用密钥文件。数字证书引入了可信赖第三方,使得签名更加真实、可靠、规范。

2 OpenType字库相关表结构

OpenType字库软件由一个字体文件构成。字体文件中不仅包含各种字体所需数据而且包含很多指令,它们以表(Table)的形式存储[4]。数据主要描述字库中字符的轮廓信息,指令主要用于对全体或者单个字符的调整和替代。字库解释系统读取这些表中的数据,并解析和执行相关表中的指令,将字体轮廓信息显示出来。在字体文件开头处的表目录(Table Directory)中,存储了所有表的标识、校验和、偏移量和长度信息[5],便于对表的查找。其结构如表1[4]所示。

表1　表目录结构

数据类型	名字	描述
32比特	Tag	4字节标识符
32比特	checkSum	表的校验和
32比特	Offset	从文件开头到本表开头的偏移量
32比特	Length	表的长度

每个表都有单独的格式,其中数字签名表(DSIG)[4]包含了字库的数字签名信息,它由一个表头(Table Header)、一个或多个格式/偏移量表(Format/Offset Table)、数字签名块(Signature Block)组成。DSIG表头描述了本字库的数字签名的整体信息,如表2[4]所示。

表2　DSIG表头结构

数据类型	名字	描述
32比特	ulVersion	版本号为0x00000001
16比特	usNumSigs	表中签名的个数
16比特	usFlag	第0比特位设置为1则不能重新签名,其余位均设置为0

格式/偏移量表紧跟在表头后面,如果有多个签名的话,就会有多个格式/偏移量表,它描述了每个签名的具体算法和位置,Windows操作系统目前支持的签名对象格式为Format1和Format4,它们分别代表不同类型的OpenType字库,如表3[4]所示。

签名数据块存储签名数据本身,如表4[4]所示。

以上介绍了字库中与数字签名密切相关的表结构,这些表格中的数据由字库签名应用程序读写,将运用于数字签名的加载和验证过程。

表3　格式/偏移量表结构

数据类型	名字	描述
32比特	ulFormat	签名对象的格式和加密算法
32比特	ulLength	签名的长度(以字节为单位)
32比特	ulOffset	从本表开头开始计算的签名数据偏移量,指明了到签名数据块的偏移

表4　签名数据块结构

数据类型	名字	描述
16比特	usReserved1	保留字
16比特	usReserved2	保留字
32比特	cbSignature	本签名的长度
字节数组	bSignature	签名数据

3 字库数字签名的实现

3.1 准备工作

首先,需要申请并获得CA颁发的数字证书,包括一个.spc文件和一个.pvk文件,它们符合PKCS#7公钥密码标准。其中,.spc文件包含公共密钥和其他信息,存储在硬盘上,可以分发给其他人;.pvk文件包含一个与公钥匹配的密钥,存储在移动设备上。这些文件也可以使用Frame SDK中的工具生成临时版本,用于程序代码的测试。其次,一个可以签名的字库必须满足十个前提条件[4],它们分别是:

1) 在head表中的magic数字必须正确。

2) 如果在offset表中给出了表的个数,那么在off-

set 表中的其他取值必须一致。

3) 表目录中的标识隐含了指向每个表的指针,必须以字母排序并且没有重复。

4) 每个表的偏移量是 4 的倍数,不足 4 的倍数则需要填充。

5) 在表之间的填充字节设置为 0。

6) 第一个实际的表必须直接跟在表目录的后面。

7) 如果表按照偏移量升序排序,那么对于所有的表 i(表的编号)必须满足以下表达式:

$$\text{offset}[i] + \text{length}[i] \leqslant \text{offset}[i+1] \quad (1)$$

$$\text{offset}[i] + \text{length}[i] \geqslant \text{offset}[i+1] - 3 \quad (2)$$

式中的 offset[i] 是编号为 i 的表相对文件开始的偏移量,length[i] 是编号为 i 的表的长度。也就是说,表之间不能重叠而且填充不能超过 3 个字节。

8) 最后一个表的偏移量加长度不能超过整个文件的大小。

9) 所有表的校验和必须正确。

10) 在 head 表中提供的整个文件的校验和必须正确。

只有在以上 10 个条件全部满足的前提下,字库才能建立签名,否则将报错。这样做的目的是为了验证字库的正确性,保证字库的质量。

3.2 具体实现

Microsoft 公司提供了一系列专为字库设计开发的数字签名应用程序,其中包括加密算法 RSA,两个哈希算法 SHA-1 和 MD5 可选其一。使用这些工具可以方便地签名一个字库或者编写少量批处理程序签名多个字库。签名一个字库的操作步骤如下:

1) 注册 mssipotf.dll。

2) 使用 signcode 命令签名字库文件,并加入时间戳标记,可选加入字库名称、厂商 web 地址。

3) 使用 chktrust 命令检验签名正确性。如果签名成功,在 windows 2000 和 windows 98 系统下,右键单击该字库文件,其属性中会出现数字签名 Tab 键,包括了证书的详细信息及其时间戳。

其中 signcode 命令内部的详细操作流程[4]为:

(1) 检查字体文件是否满足前提条件。

(2) 检查字体文件,如果已经存在 DSIG 表,那么,删除这个 DSIG 表,并且从表目录中删除 DSIG 表的入口记录,调整删除发生后变化的表格的偏移量,重新计算 head 表中的文件校验和。

(3) 使用哈希算法(SHA-1/MD5)对整个字体文件做哈希运算,产生摘要。

(4) 对鉴别标识采用 RSA 加密算法运算,产生符合 PKCS#7 标准的数字签名。

(5) 新建一个 DSIG 表,并将数字签名写入。

(6) 将 DSIG 表添加到字体文件中,适当调整偏移量。

(7) 在表目录中添加 DSIG 表的入口记录。

重新计算字体文件的校验和,并写入 head 表中。计算过程如图 1 所示。

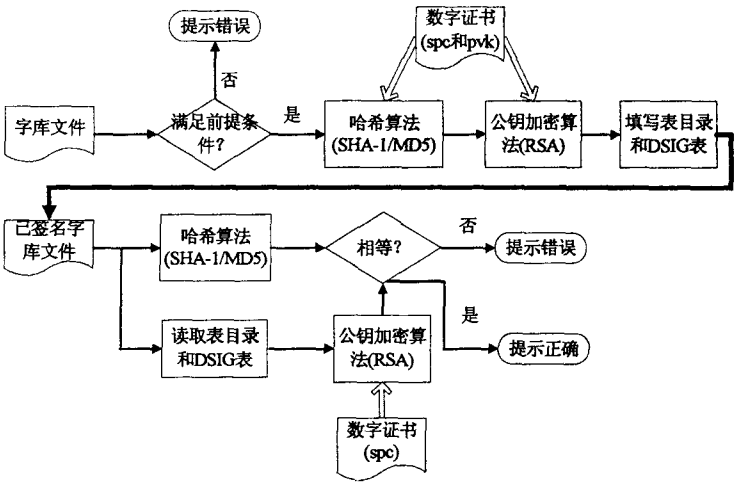

图 1 字库签名实现及其验证过程

综上所述,签名成功的字库将具有证书的详细属性。另外,如果从网上下载了签名的字库进行安装时,操作系统都会提示该字库的认证信息,用户可以阅读详细的字库身份说明,选择是否信任并且安装。

3.3 数字签名的识别

操作系统验证数字签名分为四步:

1) 读取表目录,获得 DSIG 表的位置,找到 DSIG 表。

2) 依次读取 DSIG 表头和格式/偏移量表,获得加密算法和数字签名数据。

3) 根据获得的数字签名数据和加密算法,运用数字证书及其之中公开密钥信息,验证数字签名。

4) 反馈验证结果。

4 结束语

基于数字证书技术的数字签名为用户提供了更多一层的安全保障。文中介绍的基于数字证书的数字签名技术在 OpenType 矢量字库上的实现方法,已经应用于矢量汉字字库生产,极大地提高了字库的安全性和可信性。

但是,数字签名并不是万能良药,它虽然具有很多优点,仍然不可避免地存在盲区。数字签名不能阻止字库软件的随意拷贝和在网络上的广泛传播。而且具

(下转第 25 页)

计算 $I(D \mid C)$[7]:首先列出 $\mid X_i \cap Y_j \mid = \{\mid X_1 \cap Y_j, \mid X_2 \cap Y_2 \mid, \cdots, \mid X_n \cap Y_m \mid\}$,加入方程进行计算。

③ 计算不是核的条件属性对决策属性的依赖度;// 如何计算 $\gamma_c(D)$,$U/\mathrm{IND}(C) = \{X_1, X_2, \cdots, X_n\}$,$U/\mathrm{IND}(D) = \{Y_1, Y_2, \cdots, Y_m\}$,计算 $X_i \subseteq Y_j$ 的个数 $\mid X_i \subseteq Y_j \mid$,即正区域 $\mathrm{POS}_C(D)$,$\gamma_c(D) = \dfrac{\mathrm{card}(\mathrm{POS}_C(D))}{\mathrm{card}(C)}$

④ 计算 $I(D \mid \mathrm{core}(C,D))$//这是核条件属性与决策属性的互信息量,添加新的条件属性就修正 $I(D \mid \mathrm{core}(C,D))$[8]

⑤ 如果 $I(D \mid \mathrm{core}(C,D)) = I(D \mid C)$,说明它们有同等的分类能力,结束计算,否则,将属性依赖度大的条件属性加入核条件属性表;

⑥ 重复步骤⑤,直到结束。

4　算法的测试

用 VC++6.0 开发软件实现了启发式属性约简算法和动态条件属性约简算法的平台。所有的数据采用文本文件本的型式保存,最后的结果除了在 VC 结果环境下可视以外,也保存在文本文件中,目的是为了进一步做规则分类提取算法。使算法具有更好的容错性,以及针对海量数据的特点,采用了动态存储方式。将 HSV 和 Iris 两种数据用记事本保存,对两种算法进行了测试,两个算法分别对 HSV 和 Iris 数据约简表一样。两种算法的时间复杂度和空间复杂度如表 1 所示。r 是数据记录行数,$m(m = \mathrm{column})$ 是每次的比较次数,a、b 代表每次的比较次数。

动态条件属性约简算法刚开始可以不采用差分矩阵来求核条件属性,这里用差分矩阵的目的是为进一步的规则分类做铺垫。互信息的公式计算可以采用迭代[8]的方式计算,可以保留上一次的信息,只计算新增加的信息,这里保留原始的计算在差分矩阵中差别,差别取反就是两两记录的交集。这样可节约重复的计算时间。

5 结束语

在启发式属性约简算法的基础上,提出了动态条件属性约简算法,并在 VC++6.0 环境下实现了两个算法,动态条件属性的算法是利用了新增加的条件属性列对原有的互信息进行修正。实验证明算法是实用有效的。

表 1　两种算法的比较

	启发式属性约简算法	动态属性约简算法
空间复杂度	$O(\mid C \mid)$	$O(\mid C \mid)$
时间复杂度	$r(\frac{r}{2}-1)m^2 + 2X\mathrm{br}(\frac{r}{2}-1)$	$r(\frac{r}{2}-1)m + X\mathrm{ar}(\frac{r}{2}-1)$

参考文献:

[1] Pawlak Z. Rough set[J]. International Journal of Information and Computer Science, 1982(11):341-356.

[2] Bazan J G, Skowron A, Synak P. Dynamic reducts as a tool for extracting lows from decisions tables[M]//Ras Z W, Zemankiva M. Methodologies for Intelligent Systems. Berlin: Springer-Verlag, 1994:346-355.

[3] 韩　斌,吴铁军,杨明晖.结合粗集理论的动态属性约简研究[J].系统工程理论与实践,2002(6):67-73.

[4] 刘振华,刘三阳,王　珏.基于信息量的一种条件属性约简算法[J].西安电子科技大学学报:自然科学版,2003(6):834-838.

[5] 彭黎黎,刘　山.基于信息量的动态属性约简[J].计算机工程,2005(7):104-105.

[6] 杜晓昕,徐　慧,任长伟,等.基于粗糙集的属性约简在数据挖掘中的应用[EB/OL].2006.中国科技论文在线.http://www.paper.edu.cn.

[7] 王加阳,陈松乔,罗　安.粗集动态约简研究[J].小型微型计算机系统,2006(11):2056-2060.

[8] 刘　山,张　慧.基于条件信息量的动态属性约简方法[J].计算机工程,2007(6):182-183.

(上接第 22 页)

备字库开发能力的人也能够通过技术手段从字库中删除签名部分,再修改字库,甚至重新对字库进行签名。加密和压缩可以解决上述问题,防止对字库的再次签名和修改,但这仍然有待于未来 Windows 操作系统的支持。

参考文献:

[1] Stallings W. 密码编码学与网络安全——原理与实践[M].第 3 版.刘玉珍,王丽娜,傅建明译.北京:电子工业出版社,2004.

[2] 贺卫红,曹　毅.RSA 公钥密码体制在数字签名中的应用[J].微机发展,2003,13(9):49-52.

[3] 屈喜龙.基于数字证书的数字签名系统的设计与实现[J].计算机工程与应用,2006,42(15):189-192.

[4] Microsoft Corporation. OpenType specification v.1.2 [DB/OL]. 2002. http://www.microsoft.com/typography/otspec/otff.htm.

[5] Microsoft Corporation. TrueType 字型核心技术[M].北京:学苑出版社,1993.