文章编号: 1001 - 9081(2010)05 - 1236 - 03

基于 eCryptfs的多用户加密文件系统设计和实现

唐晓东, 付松龄, 何连跃
(国防科学技术大学 计算机学院,长沙 410073)
(xdtang@nudt.edu.cn)

摘 要:加密文件系统 eCryptfs能有效防止存储介质在丢失或失窃情况下导致的信息泄露隐患。它的不足在于:不支持多用户环境下的联机数据保护,即一个用户一旦成功授权访问加密文件系统中的数据,则系统中的其他用户都可以访问加密文件系统中的数据。提出了以密钥为权能的加密文件系统访问控制机制,并基于 eCryptfs设计和实现了多用户加密文件系统,确保只有拥有合法密钥的用户才能访问密文数据。

关键词:加密文件系统;文件加密密钥;密钥加密密钥;访问控制;权能
中图分类号: TP309 **文献标志码**: A

Design and implementation of multi-user encryption file system based on eCryptfs

TANG Xiao-dong, FU Song-ling, HE Lian-yue
(*School of Computer Science, National University of Defense Technology, Changsha Hunan* 410073, *China*)

Abstract: Encryption file system such as eCryptfs can prevent the leakage of information when storage media is lost or stolen. But it can not guarantee the data security when many users online access the system at the same time, because once one user can access the encryption data, the other users can access the data too. A capability mechanism based on keys was put forward to solve the problem, and then the method was used to design and implement a multi-user encryption file system based on eCryptfs to ensure that only those users with keys can access the encryption data.

Key words: encryption file system; File Encryption Key (FEK); Key Encryption Key (KEK); access control; capability

0 引言

存储安全被认为是信息安全的最后一道防线,它主要采用密码技术,将受保护的数据以密文形式存储在介质上。这样,即使存储介质丢失或失窃了,非法者因为没有合法的密钥,不能获得信息的真实内容,从而有效保护了信息的保密性。

加密文件系统是存储安全的一种重要形式,相对其他形式来说,它具有很强的吸引力,主要体现在两个方面: 1)访问透明,用户和应用完全没有意识到数据被加密存储,数据的加解密自动完成,不需要用户的干预,非常方便; 2)安全,以文件为单位进行加密和解密,不同的文件使用不同的密钥,如果某个文件的密钥被攻击者获得,他最多只能访问这个文件,其他加密文件的保密性并没有受到破坏。

主流的操作系统都有自己的加密文件系统,如 Windows上的 efs[1-2], freebsd上的 cryptfs[3], Linux上的 eCryptfs[4]。在实际的使用过程中,用户对加密文件系统的脱机数据保护措施满意,即存储介质丢失不会导致敏感信息的泄漏;同时又感觉到加密文件系统的应用环境有限,只适合单用户系统,不适合多用户系统。原因在于,在多用户联机环境下,只要有一个用户能够访问加密文件系统中的数据,系统中的其他用户就自然地也可以访问这些数据,这就可能造成安全问题。虽然用户可以通过操作系统提供的其他安全机制对敏感数据进行保护,比如自主访问控制机制等,但自主访问控制的安全性不强,并且它对系统的超级用户 root没有任何限制,即任何用户的敏感信息对 root用户来说,毫无保密性和隐私性。这对用户来说是不可接受的。

本文针对加密文件系统在上述方面的不足,提出了以密钥为权能的访问控制机制,能够有效地支持多用户联机环境下敏感数据的保护,并在 eCryptfs基础上设计和实现了多用户加密文件系统。

1 eCryptfs

eCryptfs是一个堆栈式加密文件系统,它只处理数据的加密和解密,数据的存储交给下层文件系统处理,如 ext3、JFS等,实现相对简单。它还具有其他特性: 1)易于部署,以模块形式加载,不需要修改内核其他部分; 2)易于使用,用户执行 mount命令后,就可以透明地使用加解密存储服务; 3)兼容性好,文件的加密元数据与文件数据一起存储,存储在文件的头部,这样加密文件可以像普通文件一样进行增量备份。

1.1 文件系统结构

应用程序完全不感知加密文件系统的存在,它仍旧使用原有系统调用来进行文件相关操作。

eCryptfs介于虚拟文件系统层和下层文件系统之间,调用内核密码接口对数据进行加解密。

内核密钥存储组件和用户层的 eCryptfs守护进程一起对 eCryptfs使用的密钥进行维护。

收稿日期: 2009 - 10 - 22。 **基金项目**:国家 863计划项目 (2007AA01Z408)。
作者简介:唐晓东 (1973 -),男,湖南长沙人,副研究员,硕士,主要研究方向:信息安全、系统软件; 付松龄 (1978 -),男,四川眉山人,博士研究生,主要研究方向:信息安全、系统软件; 何连跃 (1971 -),男,浙江金华人,副研究员,博士,主要研究方向:信息安全、系统软件。

1.2 密钥管理

eCryptfs使用对称密钥加密算法,如 DES3、AES-128,对文件数据进行加密,文件加密密钥(File Encryption Key, FEK)是随机产生的,一个文件对应一个密钥。

FEK是敏感信息,不能以明文形式存储,eCryptfs采用认证特征相关的密钥加密 FEK,称认证特征相关的密钥为密钥加密密钥(Key Encryption Key, KEK),加密后的 FEK称为 EFEK。eCryptfs支持两种认证特征:口令字和公钥。

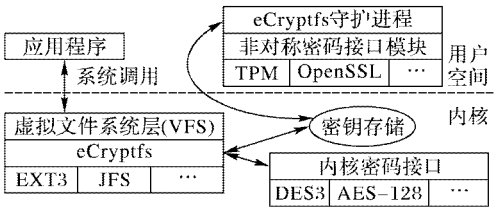


图 1　eCryptfs文件系统结构

1.3 vfs操作

eCryptfs主要与下列虚拟文件系统(Virtual File System, VFS)操作相关。

mount操作　挂载时,需要指定认证特征参数,操作成功后,这个认证特征将被存储在内核密钥环中,有两个用途。1)充当密钥加密密钥 KEK,用于加密 FEK。2)在解密 EFEK时,用于关联密钥解密密钥:如果认证特征为口令字,则密钥解密密钥就是 KEK;如果认证特征为公钥,则密钥解密密钥是公钥对应的私钥。

open操作　如果文件已存在,则请求下层文件系统从文件头部读取 EFEK,然后请求解密以获取 FEK,把它保存在 inode节点中。如果文件不存在,则随机生成 FEK,用 KEK加密生成 EFEK,保存到文件头部。

读操作　用 FEK解密从存储介质读取的文件数据。

写操作　用 FEK加密数据,并写入存储介质。

1.4 不足分析

假设目录 /ecrypt为 eCryptfs加密文件系统根目录,当未正常挂载时,用户打开该目录下的文件所看到的是一堆乱码,无法获取文件的真实内容。但是当某个用户使用 mount命令并提供正确的认证特征挂载了该系统时,不但他可以看到文件的真实内容,系统的其他用户也可以看到文件的真实内容。

为了弥补这个缺陷,eCryptfs建议用户使用操作系统的其他安全机制,比如自主访问控制,对文件的访问进行按需授权。基于自主访问控制,客体的拥有者可以自主地对其他用户或用户组进行访问授权,能直接或间接地将访问权授予其他主体。但是由于自主访问控制授权的自主性和操作系统中存在着大量不可信主体,很难杜绝非意愿授权的存在,从而导致非法用户可以浏览到敏感信息。即便自主访问控制能够确保所有授权都是按意愿授权,仍无法回避的一个不足就是不能控制超级用户 root的访问,因为超级用户不受自主访问控制的限制,因此他能够浏览所有用户的信息。一般情况下,用户认为既然使用加密方式存储信息就是因为信息非常敏感或隐私,希望这些信息只能自己访问,其他用户不能访问。

经过对 eCryptfs源代码的深入分析,发现 eCryptfs不支持多用户环境下的联机数据保护的原因在于:当 mount文件系统时,系统将认证特征相关联的密钥加密密钥 KEK存储到内核密钥环后,没有对其进行必要的保护,导致系统中的任何主体/进程都可以使用这个密钥,从而获取密文内容。为了解决上述安全问题,本文引入了以密钥为权能的访问控制机制,通过对 KEK实施授权访问,最终达到对加密数据进行授权访问的目的。以下章节中,如果没有显示指出,密钥缺省指的是 KEK。

2 密钥权能访问控制

密钥权能访问控制是一种能力机制。能力机制是从主体的角度描述访问控制,一个最大优点是便于实施最小特权原则,大大降低超级用户带来的风险。用户只有具有相应密钥权能才能访问加密文件系统,如果超级用户没有获取相应密钥权能,他也不能访问加密文件系统中的数据。

2.1 密钥权能

在操作系统中,能力表现为特权,按照 IEEE POSIX 1003.1e[5]标准,操作系统中的特权可划分为挂载文件系统、执行网络管理任务、加载内核模块和数据备份等多种,总数相对稳定。

在 eCryptfs加密文件系统中,能力体现为拥有密钥,即密钥权能。一个加密文件系统对应一个密钥,系统中加密文件系统数目是变化的,密钥权能的数目也随着变化。假设当前系统中存在 m 个 eCryptfs加密系统,它们的密钥分别表示为 $key_1, key_2, \cdots, key_m$,则相应的密钥权能记为 $C_{key_1}, C_{key_2}, \cdots, C_{key_m}$。密钥权能全集可表示为:

$$\Sigma_c = \{C_{key_1}, C_{key_2}, \cdots, C_{key_m}\} \tag{1}$$

2.2 主体能力

假设操作系统中,存在 n 个进程,记为:$P_1, P_2, \cdots, P_n$。Σ_p 表示系统中所有进程的集合。每个进程的密钥权能都对应着 Σ_c 的一个子集,函数定义如下:

$$CapProc: \Sigma_p \rightarrow 2^{\Sigma_c} \tag{2}$$

主体能力描述了主体能够访问哪些加密文件系统。缺省情况下,进程的密钥权能为空,并且需要显式操作才能改变进程的密钥权能。

2.3 权能维护

权能维护涉及到权能的改变和权能在主体间的传递。

2.3.1 权能更改

进程密钥权能的改变需要显式的操作才能完成,式(3)定义了该操作 $AddKey$。

$$AddKey(P_i, key_j) \Leftrightarrow CapProc(P_i) = CapProc(P_i) + \{C_{key_j}\};\ 0 < i \leq n, 0 < j \leq m \tag{3}$$

$AddKey$操作是修改进程密钥权能的唯一途径。操作成功后导致进程 P_i 的密钥权能在原有的基础上新增了密钥 key_i 对应的权能 C_{key_i}。

显然针对每个进程都显示执行 $AddKey$操作,非常繁琐,与 eCryptfs简单易用、对用户透明的设计初衷相违背,用户不会接受。

为此,本文使用会话作为 $AddKey$操作的对象。在 UNIX系统中,系统由多个会话组成,会话由一个或多个进程组组成,进程组由一个或多个进程组成。正常情况下,系统中的会话数要远远低于进程数。假设系统中存在 r 个会话,分别为 $S_1, S_2, \cdots, S_r$。Σ_S 表示系统中所有会话集合。Σ_{S_i} 描述属于会话 S_i 的进程集合。用会话代替进程作为密钥权能管理的单位,相关公式表述如下:

$$CapSess: \Sigma_S \rightarrow 2^{\Sigma_c} \tag{4}$$

$$CapSess(S_i) = \sum CapProc(p);\ 0 < i \leq r, p \in \Sigma_{S_i} \tag{5}$$

$$AddKey(S_i, key_j) \Leftrightarrow CapSess(S_i) = CapSess(S_i) + \{C_{key_j}\};\ 0 < i \leq r, 0 < j \leq m \tag{6}$$

$$AddKey(S_i, key_j) \Rightarrow \forall p(AddKey(p, key_j)),\ p \in \Sigma_{S_i},\ 0 < i \leq r, 0 < j \leq m \quad (7)$$

函数 $CapSess$描述会话的密钥权能对应着 Σ_c的一个子集。式 (5) 说明会话的密钥权能是所属会话中所有进程的密钥权能和。式 (6) 说明同样通过 $AddKey$操作修改会话的密钥权能,操作成功后将导致会话的密钥权能在原有的基础上新增添加的权能。式 (7) 表明对会话进行 $AddKey$操作的结果相当于会话中的所有进程都执行 $AddKey$操作,增加与会话相同的权能。

2.3.2 权能传递

密钥权能依附于主体,密钥权能在主体间传递仅发生在创建新的主体时,主要涉及到两个操作 $fork$和 $setsid$。$fork$创建新进程,假设 S_{parent}是父进程, S_{child}是子进程。$setsid$创建新会话,假设 $NSess$是新会话, $OSess$是旧会话。权能传递公式定义如下:

$$CapProc(S_{child}) = CapProc(S_{parent}) \quad (8)$$

$$CapSess(NSess) = CapSess(OSess) \quad (9)$$

式 (8)、(9) 表明创建新主体时,新主体仅是简单地从旧主体中继承所有权能。

综合式 (7) ~ (9),可以得出以下结论:

$$\forall s_1\ \forall s_2\ (CapProc(s_1) = CapProc(s_2) = CapSess(S_i));\ s_1, s_2 \in \Sigma_{S_i},\ 0 < i \leq r \quad (10)$$

$$(CapSess(S_i) = CapSess(S_j)) \&\& (AddKey(S_i, key_x) \| AddKey(S_j, key_y)) \Rightarrow CapSess(S_i) \neq CapSess(S_j);\ i \neq j,\ x \neq y,\ 0 < i \leq r,\ 0 < j \leq r,\ 0 < i \leq m,\ 0 < y \leq m \quad (11)$$

式 (10)描述了属于同一个会话的任何两个进程,它们的密钥权能相等,都等于会话的密钥权能。既然普通用户会话中的所有进程都属于同一个用户,该设计不与多用户加密文件系统的设计目标相违背,因此用会话密钥权能来描述会话中的进程密钥权能。

式 (11)描述了对于系统中存在的任意两个会话 S_i和 S_j,即使某个时间,它们的密钥权能相等,如果之后,对其中的一个会话显示改变权能,或者对两个会话分别显示添加不能的权能,则两个会话的权能将不再一样。这就为多用户加密文件系统实现联机数据保护提供了保证,针对合法用户,为其会话增加相应的密钥权能,这个密钥权能不会传递给其他会话,确保只有合法用户能访问,非授权用户不能访问。

3 多用户加密文件系统设计

3.1 AddKey设计

AddKey主要功能是将密钥权能赋予会话。为了安全性,在赋予会话之前必须验证密钥的合法性。图 2描述了其工作流程,支持 eCryptfs的两种认证特征。

当认证特征为口令字时,为了增强安全强度,不是直接将用户输入的口令字作为 KEK,而是使用盐值与输入的口令字进行连接,然后进行 Hash计算得到 KEK。

验证公钥合法性主要是通过挑战响应协议,验证用户拥有与公钥匹配的私钥。流程如下:

1)系统产生随机数,请求用户签名;

2)用户使用私钥进行签名,将签名返回给系统;

3)系统使用公钥对签名进行验证;

4)通过验证则公钥合法,否则不合法。

基于性能考虑,对公钥进行必要的变换后,把它作为密钥加密密钥 KEK,方便权能验证。

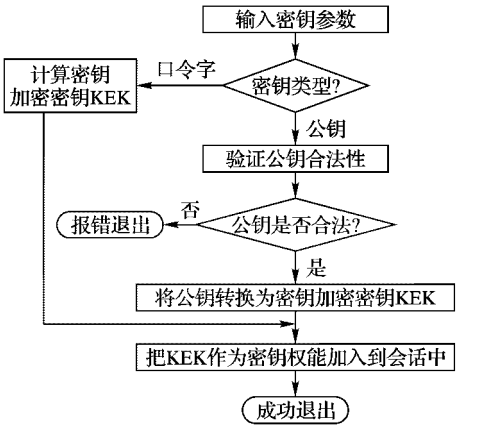


图 2 AddKey工作流程

3.2 权能验证设计

权能验证用于验证主体是否拥有合法的密钥权能,为访问控制授权提供凭证。式 (12)描述了授权规则, C_{FS}表示访问文件系统 FS需要的权能, $Grant(p, FS)$表示授予主体 p访问文件系统 FS,当进程访问加密文件系统时,只要进程所属会话的密钥权能集包含文件系统对应的密钥权能,则授权进程访问,其他情况下拒绝进程访问。

$$C_{FS} \subseteq CapSess(S_i) \Rightarrow Grant(p, FS);\ p \in \Sigma_{S_i},\ 0 < i \leq r \quad (12)$$

大多数情况下, C_{FS}只有一个元素。

4 多用户加密文件系统实现

实现多用户加密文件系统主要在 eCryptfs原有基础上增加两方面功能: 1)支持 AddKey; 2)权能验证。

4.1 AddKey实现

系统提供两种途径实现 AddKey功能: pam模块和命令行工具。pam模块名为 pam_MultiUsereCryptfs so,命令行工具执行文件名为 SessAddKey。

pam_MultiUsereCryptfs so用于支持 pam的应用程序,比如 ftp、login、ssh等。下面以 login为例进行说明,需要在 /etc/pam.d/login文件中增加下列一行信息。

```
Session optional /lib/security/pam_MultiUsereCryptfs so
    /etc/authtoken.conf
```

命令行工具 SessAddKey用于手工为会话增加密钥权能,命令格式如下:

```
SessAddKey [-s sid] /etc/authtoken.conf
```

选项“-s sid”是可选的,用于显示指定会话标识,如果没有指定,则缺省为当前会话。

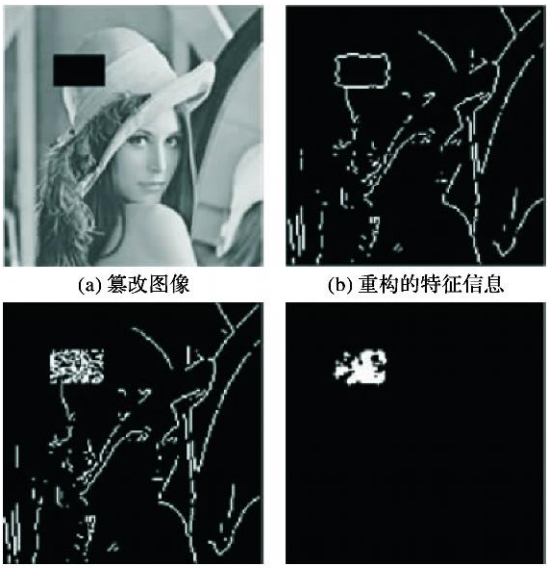
authtoken.conf是配置文件,提供认证特征相关的信息。格式如下:

```
#认证特征配置文件                          /注释行
#[PASSWORD]                    /设置口令字认证特征
[PUBKEY]                      /设置公钥认证特征
filename                      /公钥文件名
```

上述配置文件采用的是公钥认证特征,公钥由文件名 filename指定。如果需要配置口令字认证特征,在上述配置文件中取消第二行的注释符号,同时注释掉公钥认证特征相关信息。采用口令字认证特征时,在 pam模块执行过程中将提示用户输入口令字。如果配置文件不存在或配置信息为空,则采用口令字认证特征,口令字为用户认证口令。

(下转第 1242页)

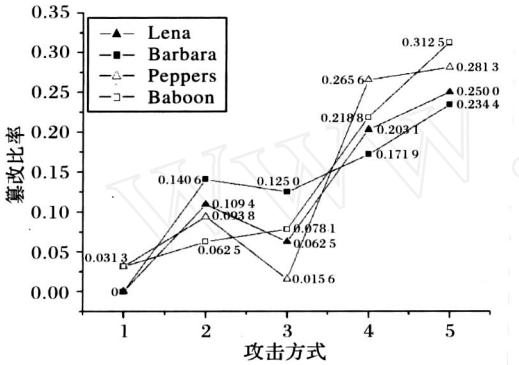
算法不仅能够提高恶意处理的检测精度,而且可以增强常规操作的免疫性能。



(a) 篡改图像 (b) 重构的特征信息

(c) 解调后的特征信息 (d) 篡改定位

图 6 剪切 Lena图后的篡改检测结果



注:横坐标中1表示JPEG压缩90%;2表示JPEG压缩80%;3表示椒盐噪声1%;4表示高斯噪声σ^2=30;5表示中值滤波4×4。

图 7 常规操作下篡改比率 $\widetilde{R}^*$ 的测试结果

4 结语

本文在研究图像认证水印技术的基础上,提出了一种面向图像内容认证的半脆弱数字水印算法。该算法以小波变换为基础,结合混沌理论与非对称加密体制,构造基于图像内容的数字水印。嵌入和认证时分别采用了私钥和公钥,拥有公钥的用户在对图像进行认证后,因缺少私钥而不能再对图像进行伪造,提高了数字水印的安全性和实用性;采用视觉感知模型来量化嵌入水印,提高了数字水印的透明性。实验结果表明,该算法对常规操作免疫,对恶意处理能够准确判断并篡改定位。

参考文献:

[1] LU C-S, LIAO H-F M. Multipurpose watermarking for image authentication and protection [J]. IEEE Transactions on Image Processing, 2001, 10(10): 1579 - 1592.

[2] WONG P W. A public key watermark for image verification and authentication [C]// ICIP 98: Proceedings of the IEEE International Conference on Image Processing. Washington, DC: IEEE Press, 1998, 1: 455 - 459.

[3] CHANG C-C, LIN P-Y. Adaptive watermark mechanism for rightful ownership protection [J]. Journal of Systems and Software, 2008, 81(7): 1118 - 1129.

[4] 李剑, 李生红, 孙锬锋. 基于 Logistic混沌序列和奇异值分解的半脆弱水印算法 [J]. 上海交通大学学报, 2009, 43(7): 1144 - 1154.

[5] QI HUIYAN, ZHENG DONG, ZHAO JIYING. Human visual system based adaptive digital image watermarking [J]. Signal Processing, 2008, 88(1): 174 - 188.

[6] SUN Q B IN, CHANG S-F, KURATO M, *et al*. A quantitative semi-fragile JPEG2000 image authentication system [C]// ICIP 02: Proceedings of the IEEE International Conference on Image Processing. Washington, DC: IEEE Press, 2002, 2: 921 - 924.

[7] 王振飞,施保昌,王能超. 基于小波变换和人类视觉系统的稳健水印算法 [J]. 华中科技大学学报:自然科学版, 2007, 35(1): 26 - 28.

[8] 王向阳, 杨红颖, 侯丽敏. 一种新的半脆弱彩色图像数字水印算法 [J]. 自动化学报, 2007, 33(6): 561 - 566.

[9] 刘九芬, 黄达人, 胡军全. 数字水印中的正交小波基 [J]. 电子与信息学报, 2003, 25(4): 453 - 459.

[10] 王国栋, 刘粉林, 刘媛, 等. 一种能区分水印或内容篡改的脆弱水印算法 [J]. 电子学报, 2008, 36(7): 1349 - 1354.

[11] SUN RUI, SUN HONG, YAO TIANREN. A SVD- and quantization based semi-fragile watermarking technique for image authentication [C]// Proceedings of the 6th International Conference on Signal Processing. Washington, DC: IEEE Press, 2002, 2: 1592 - 1595.

(上接第 1238页)

4.2 权能验证实现

实现权能验证钩子函数 check_key_capability放置在 eCryptfs文件系统的相应位置,比如 Open操作,对所有访问内核密钥环的请求进行检测,达到只有合法用户能够访问的目的。

5 结语

存储安全在信息安全中扮演着越来越重要的角色。加密文件系统像其他存储安全机制一样能出色地完成脱机情况下的数据保护,不足在于:在多用户环境下,不能提供联机的数据保护。本文分析了造成 eCryptfs这种不足的主要原因,指出了采用自主访问控制解决方法的不足,提出了基于密钥权能的访问机制,设计和实现了多用户加密文件系统,满足多用户环境下联机数据保护的目的,即使是系统的超级用户在联机环境下也无法看到用户的密文数据,要访问密文数据必须提供合法的密钥权能。

参考文献:

[1] 徐国栋,白英彩. 加密文件系统在 Windows下的实现 [J]. 微型电脑应用, 2006, 22(5): 56 - 58.

[2] 沈士根. EFS的研究与安全性分析 [J]. 微计算机信息, 2006, 22(24): 96 - 98.

[3] ZADOK E, BADULESCU I, SHENDER A. Cryptfs: A stackable vnode level encryption file system, CUCS-021-98 [R]. New York: Columbia University, Computer Science Department, 1998.

[4] HALCROW M A. eCryptfs: An enterprise-class cryptographic filesystem for Linux [EB/OL]. [2009 - 08 - 22]. http://www.dubeyko.com/development/FileSystems/eCryptfs/ecryptfs.pdf.

[5] IEEE/ANSI Draft Std. 1003.1e. Draft standard for information technology - POSIX Part 1: System API: Protection, audit and control interface [S]. IEEE, 1997.