(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 107111726 A
(43)申请公布日 2017.08.29

(21)申请号 201680006611.4

(22)申请日 2016.01.20

(30)优先权数据
62/105632 2015.01.20 US
62/171708 2015.06.05 US
14/957789 2015.12.03 US

(85)PCT国际申请进入国家阶段日
2017.07.20

(86)PCT国际申请的申请数据
PCT/US2016/014053 2016.01.20

(87)PCT国际申请的公布数据
WO2016/118580 EN 2016.07.28

(71)申请人 微软技术许可有限责任公司
地址 美国华盛顿州

(72)发明人 D.欧阳 P.诺沃特尼 R.欣德

(74)专利代理机构 中国专利代理(香港)有限公司 72001
代理人 李舒 陈岚

(51)Int.Cl.
G06F 21/62(2013.01)

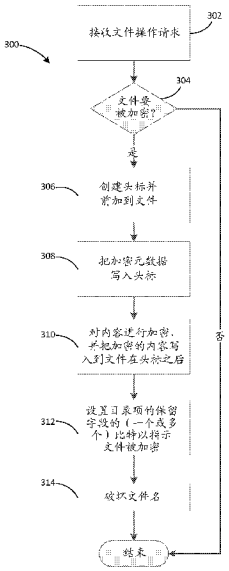
权利要求书2页 说明书16页 附图4页

(54)发明名称

对FAT文件系统的文件加密支持

(57)摘要

本机文件加密支持被集成到不提供此类支持的现有文件系统中，例如FAT系的文件系统，而同时维持与这些文件系统的先前实现的后向兼容性。



CN 107111726 A

1.一种计算设备，其包括处理单元、存储器和在所述处理单元上执行的文件系统，当所述文件系统在所述处理单元上执行时，其完成以下操作，包括：

响应于指示对文件进行加密的请求的文件操作，给所述文件前加一个头标；

加密所述文件的内容，并将加密的内容写入所述文件中所述头标之后；

将与该加密文件相关的加密元数据存储在所述头标内；

在用于所述文件的文件目录项的保留字段中指示所述文件被加密；和

将额外的文件名扩展名附加于该加密文件的文件名，进一步指示所述文件被加密。

2.如权利要求1中所述的计算设备，其中所述文件系统设置所述保留字段的单个比特来指示所述文件被加密。

3.如权利要求1或 2的任一项中所述的计算设备，其中所述前加的头标具有一个初始的预定大小，其可以由所述文件系统进行扩展，所述文件系统用所述保留字段的第二比特来指示该前加的头标是否超过所述初始的预定大小。

4.如权利要求1至3的任一项中所述的计算设备，其中所述加密操作包括使用基于固定块大小操作的加密算法来对文件的内容进行加密，并且其中所述文件系统还执行以下操作，包括：

把所述文件的所述内容填充成与最后的加密块的块大小调准；和

把填充量的指示存储在用于所述文件的对于文件目录项的保留字段中。

5.如权利要求4中所述的计算设备，其中所述文件系统还在前加的头标中存储以下的至少一项：(a) 该加密文件的真实大小的指示或 (b)该填充量的指示。

6.如权利要求1至5的任一项中所述的计算设备，其中所述文件系统还执行以下操作，包括：

检测如下的文件，即：其具有的文件名带有指示所述文件是加密文件的额外的已附加的扩展名、但其中所述文件目录的保留字段并未指示所述文件被加密：

读取至少一些文件内容，以确定开头是否被证实为其中存储有加密元数据的前加的头标；和

如果所述至少一些文件内容被证实为前加的头标，则进一步执行以下的一项或多项：

(a) 将高速缓存的结果留在存储器中，以提高对于这个目录项的未来效率；

(b) 如果其上存储所述文件的存储介质是可写的，则更新用于所述文件的目录项的保留字段，以指示所述文件被加密；和

(c) 向用户隐藏该额外的扩展名，并报告未加密数据的真实大小。

7.如权利要求1至6的任一项中所述的计算设备，其中所述文件系统还执行以下操作，包括：

通过以下方式将该加密文件从第一目录移动到第二目录：

在第二目录中分配一个新的目录项；

从第一目录中读取所述目录项的内容；

将所述内容写入第二目录的所述新目录项；和

将第一目录的所述目录项标记为未使用。

8.一种计算设备，其包括处理单元、存储器和在所述处理单元上执行的文件系统，当所述文件系统在所述处理单元上执行时，其完成以下操作，包括：

响应于指示对文件进行加密的请求的文件操作，给所述文件前加一个头标；

加密所述文件的内容，并将加密的内容写入所述文件中；

将与该加密文件相关的加密元数据存储在所述头标内；

为所述文件创建额外的文件目录项，其跟随在文件目录中的用于所述文件的其他目录项之后；

将关于所述文件的真实大小的信息存储在所述额外的文件目录项中；

对用于所述文件的其他文件目录项的至少一部分执行密码术操作，并将密码术操作的结果存储在所述额外的文件目录项中；和

将所述额外的文件目录项标记为被删除的或未使用的文件目录项。

9.如权利要求8中所述的计算设备，所述文件系统还将额外的文件名扩展名附加到所述文件的文件名。

10.如权利要求8或9的任一项中所述的计算设备，其中所述文件系统还将所述前加的头标的大小存储在所述额外的文件目录项中。

11.如权利要求8至10的任一项中所述的计算设备，其中对用于所述文件的其他文件目录项的至少一部分执行的所述密码术操作包括计算校验和。

12.如权利要求8至11的任一项中所述的计算设备，其中所述加密操作包括使用基于固定块大小操作的加密算法来加密所述文件的内容，并且所述文件系统把所述文件的内容填充成与最后的加密块的块大小调准，被存储在额外目录项中的关于所述文件的真实大小的信息反映所述文件在没有填充的情况下的大小。

13.如权利要求12中所述的计算设备，其中所述文件系统还在前加的头标中存储以下的至少一项：(a)加密文件的真实大小的指示或 (b)填充量的指示。

14.一种在计算设备的文件系统中提供本机文件加密支持的方法，包括：

响应于指示对文件进行加密的请求的文件操作，给所述文件前加一个头标；

加密所述文件的内容，并将加密的内容写入所述文件中所述头标之后；

将与该加密文件相关的加密元数据存储在所述头标内；

在用于所述文件的文件目录项的保留字段中指示所述文件被加密；和

将额外的文件名扩展名附加于所述加密文件的文件名，进一步指示所述文件被加密。

15.如权利要求14至18的任一项中所述的方法，所述保留字段包括FAT32文件系统的短目录项的NtByte字段或exFAT文件系统的FILE目录项的Reserved0字段。

对FAT文件系统的文件加密支持

[0001] 相关申请的交叉引用

本申请根据35U.S.C. §119(e) 要求 2015年1月20日提交的题为“Native File Encryption Support on FAT Family File Systems (FAT系文件系统上的本机文件加密支持)”的美国临时专利申请No.62 / 105,632和2015年6月5日提交的题为“File Encryption Support for FAT File Systems(对于FAT文件系统的文件加密支持)”的美国临时专利申请No.62/171,708的权益,并且根据35 U.S.C. § 120要求2015年12月3日提交的题为“File Encryption Support for FAT File Systems(对于FAT文件系统的文件加密支持)”的美国专利申请No.14/957,789的权益,它们的内容通过引用整体并入本文。

背景技术

[0002] 在计算中,文件系统经常是操作系统的组件,用于存储和组织计算机文件及其所包含的数据以使得易于查找和访问它们。某些文件系统,诸如新技术文件系统(NTFS),为文件级加密提供本机支持。在NTFS内是通过加密文件系统(EFS)组件来提供文件级加密支持。当文件被NTFS上的EFS加密时,EFS为该文件创建一$ EFS属性,用来存储用于该加密文件的加密元数据,诸如证书、初始化向量、文件加密密钥等。

发明内容

[0003] 本公开内容的一些方面使得本机文件加密支持能够被集成到不提供此类支持的现有文件系统中,诸如FAT系的文件系统,而同时维持与这些文件系统的先前实现的后向兼容性。本文所公开的一些方面包括用于在保持后向兼容性的同时在文件系统中存储加密元数据的方式、用于操纵文件名以使得较旧的操作系统仍然可以看到和操纵那些文件的方式、以及用于操纵文件元数据以使得文件系统的先前实现及其相关联的操作系统仍然可以解译那些文件并且在不损坏数据的情况下使用该文件系统的方式。

[0004] 本概要被提供来以简化的形式介绍概念的选择,这些概念在下面的详细说明中进一步被描述。本概要既不打算标识所要求保护的主题的关键特征或必要特征,也不打算用来限制所要求保护的主题的范围。

附图说明

[0005] 当结合附图阅读时,将更好地理解上述的概要以及下面的详细说明。为了图示本公开内容,示出了本公开内容的各个方面。然而,本公开内容不限于所讨论的特定方面。在图中:

图1图示了在其中可以采用本文公开的一些方面的示范性操作环境;

图2是图示在文件系统的目录中的用于文件的一连串目录项的图;

图3是图示用于在维持与文件系统的先前实现的后向兼容性的同时将本机加密支持集成到文件系统中的方法的一个实施例的流程图;以及

图4是图示用于在维持与文件系统的先前实现的后向兼容性的同时将本机加密支持集

成到文件系统中的方法的另一个实施例的流程图。

具体实施方式

[0006] 和对文件级加密提供本机支持的文件系统不同，其他文件系统——诸如文件分配表(FAT)系的文件系统，其包括FAT32、FAT16、FAT12和exFAT文件系统——对文件加密不提供本机支持。在这些文件系统中，没有用于存储对于加密文件的加密元数据的本机机制，诸如现有的文件流或文件属性。

[0007] FAT系的文件系统经常被使用在存储卡(SD)和其他可拆卸介质上。目前，加密SD卡或其他可拆卸介质的内容需要利用的确支持本机加密的文件系统来重新格式化整个存储介质，或是需要使用诸如BitLocker之类的加密工具。但是，这样的工具经常只提供卷级的加密，而不是文件级加密。

[0008] 本公开内容的一些方面使得本机文件加密支持能够被集成到不提供此类支持的现有文件系统中，例如FAT系的文件系统，而同时维持与这些文件系统的先前实现的后向兼容性。本文所公开的一些方面包括用于在保持后向兼容性的同时在文件系统中存储加密元数据的方式、用于操纵文件名以使得较旧的操作系统仍然可以看到和操纵那些文件的方式，以及用于操纵文件元数据以使得文件系统的先前实现及其相关联的操作系统仍然可以解译文件并且在不损坏数据的情况下使用该文件系统的方式。

[0009] 图1图示了用于实现本公开内容的各个方面的示范性环境100。如图所示，环境100包括计算设备112。计算设备112可以是各种各样不同类型的计算设备中的任何一种，包括但不限于计算机、个人计算机、服务器、便携式计算机、移动计算机、可穿戴式计算机、膝上型电脑、平板电脑、个人数字助理、智能电话、数码相机或任何其他自动执行计算的机器。

[0010] 计算设备112包括处理单元114、系统存储器116和系统总线118。系统总线118将系统组件耦合到处理单元114，系统组件包括但不限于系统存储器116。处理单元114可以是各种可用处理器的任何一种。双微处理器和其他多处理器架构也可以被采用来作为处理单元114。

[0011] 系统总线118可以是若干类型的(一个或多个)总线结构中的任何一种，包括存储器总线或存储器控制器、外围总线或外部总线、和/或使用任何的各种各样的可用总线架构的本地总线，所述总线架构包括但不限于工业标准架构(ISA)、微通道架构(MSA)、扩展的ISA(EISA)、智能驱动电子(IDE)、VESA本地总线(VLB)、外围组件互连(PCI)、卡总线、通用串行总线(USB)、高级图形端口(AGP)、个人计算机存储卡国际协会总线(PCMCIA)、火线(IEEE 1394)和小型计算机系统接口(SCSI)。

[0012] 系统存储器116包括易失性存储器120和非易失性存储器122。基本输入/输出系统(BIOS)被存储在非易失性存储器122中，其包含诸如在启动期间用于在计算设备112内的单元之间转移信息的基本例程。作为举例说明而非限制，非易失性存储器122可以包括只读存储器(ROM)、可编程ROM(PROM)、电可编程ROM(EPROM)、电可擦除ROM(SRAM)或闪速存储器。易失性存储器120包括充当外部高速缓冲存储器的随机存取存储器(RAM)。作为举例说明而不是限制，有许多形式的RAM可供使用，诸如同步RAM(SRAM)、动态RAM(DRAM)、同步DRAM(SDRAM)、双倍数据速率SDRAM(DDR SDRAM)、增强型SDRAM(ESDRAM)、同步链路DRAM(SLDRAM)和直接Rambus RAM(DRRAM)。

[0013] 计算设备112还包括可拆卸/不可拆卸的、易失性/非易失性的计算机可读存储介质。图1图示了例如盘存储装置124。盘存储装置124包括但不限于类似如下的设备：磁盘驱动器、软盘驱动器、磁带驱动器、Jaz驱动器、Zip驱动器、LS-100驱动器、存储卡(诸如SD存储卡)或记忆棒。此外，盘存储装置124可以单独地包括存储介质或与其他存储介质相组合地包括存储介质，其他存储介质包括但不限于光盘驱动器，诸如压密盘ROM设备(CD-ROM)、CD可刻录驱动器(CD-R驱动器)、CD可重写驱动器(CD-RW驱动器)或数字通用盘ROM驱动器(DVD-ROM)。为了便于将盘存储设备124连接到系统总线118，通常使用可拆卸或不可拆卸的接口，诸如接口126。

[0014] 图1还描绘了在用户和适当操作环境100中所描述的基本计算机资源之间充当中介的软件。这样的软件包括操作系统128。可以被存储在盘存储装置124上的操作系统128行动，以控制和分配计算设备112的资源。应用130通过存储在系统存储器116或盘存储装置124中的程序模块132和程序数据134而利用由操作系统128进行的对资源的管理。要理解的是，本文描述的一些方面可以用各种操作系统或操作系统的组合来实现。如进一步显示的，操作系统128包括文件系统129，其用于在盘存储装置124上存储和组织计算机文件及其包含的数据，以使得易于查找和访问它们。

[0015] 用户可以通过(一个或多个)输入设备136将命令或信息输入到计算设备112。输入设备136包括但不限于：诸如鼠标的指示设备、跟踪球、触控笔、触摸板、键盘、麦克风、操纵杆、游戏手柄、卫星碟形天线、扫描仪、电视调谐器卡、数码相机、数码摄像机、网络摄像机等。这些和其他输入设备经由(一个或多个)接口端口138通过系统总线118连接到处理单元114。(一个或多个)接口端口138包括例如串行端口、并行端口、游戏端口和通用串行总线(USB)。(一个或多个)输出设备140使用与(一个或多个)输入设备136相同类型端口的其中一些。因此，例如，可以使用USB端口来向计算设备112提供输入以及将信息从计算设备112输出到输出设备140。输出适配器142被提供来举例说明除了其它输出设备140外，尤其存在需要特殊适配器的一些输出设备140，类似监视器、扬声器和打印 机。作为举例说明而非限制，输出适配器142包括视频和声音卡，其提供在输出设备140和系统总线118之间的连接的手段。应当注意，其他的设备和/或设备的系统，比如(一个或多个)远程计算机144，同时提供输入和输出能力。

[0016] 计算设备112可以通过使用到一个或多个远程计算设备(诸如(一个或多个)远程计算设备144)的逻辑连接而在联网环境中操作。(一个或多个)远程计算设备144可以是个人计算机、服务器、路由器、网络PC、工作站、基于微处理器的器具、对等设备、与计算设备112相同的另一计算设备等等，并且通常包括相对于计算设备112描述的许多或全部单元。为简洁起见，仅把记忆存储设备146与(一个或多个)远程计算设备144一起图示出来。(一个或多个)远程计算设备144通过网络接口148被逻辑地连接到计算设备112，然后经由通信连接150被物理地连接。网络接口148包括通信网络，诸如局域网(LAN)和广域网(WAN)。LAN技术包括光纤分布式数据接口(FDDI)、铜分布式数据接口(CDDI)、以太网、令牌环等。WAN技术包括但不限于点对点链路、类似综合业务数字网(ISDN)及其变体之类的电路交换网络、分组交换网络和数字用户线路(DSL)。

[0017] (一个或多个)通信连接150指的是被采用来将网络接口148连接到总线118的硬件/软件。尽管为了直观清晰，通信连接150被显示在计算设备112的内部，但它也可以在计

算设备112的外部。仅出于示范的目的，对于连接到网络接口148来说所必需的硬件/软件包括诸如调制解调器这样的内部和外部技术，包括常见的电话级(telephone grade)调制解调器、电缆调制解调器和DSL调制解调器、ISDN适配器和以太网卡。

[0018] 当在本文中使用时，术语“组件”、“系统”、“模块”等打算是指计算机相关的实体，其或者是硬件、硬件和软件的组合、软件，或者是执行中的软件。例如，组件可以是、但不限于是在处理器上运行的进程、处理器、对象、可执行文件、执行的线程、程序和/或计算机。作为举例说明，在服务器上运行的应用和该服务器都可以是组件。一个或多个组件可以驻留在进程和/或执行的线程内，并且组件可以局限于一个计算机上和/或分布在两个或更多个计算机之间。注意，对于本文所例示的数据结构，所有字段被描述为小端模式(little endian)。

[0019] 当在本文中使用时，术语“下级(down-level)文件系统”是指不包括本文所述改进的文件系统的先前实现。相反，“上级(up-level)文件系统”是指的确包括本文所述改进的文件系统实现。

[0020] 如上所述，许多文件系统，诸如包括FAT32、FAT16、FAT12和exFAT文件系统的FAT系的文件系统，不提供对文件加密的本机支持。FAT系的文件系统经常被使用在诸如SD卡这样的存储卡和其他可拆卸介质上。绝大多数便携式计算设备用户，诸如平板电脑、智能手机和数码相机用户，预计将来会使用存储卡来存储应用和个人文件。为了提供对用户文件的隐私保护和对应用的知识产权保护，将需要加密。然而，由于大多数存储卡使用不支持本机加密的文件系统，因此加密存储卡或其他可拆卸介质的内容目前或是需要利用的确支持本机加密的文件系统(如NTFS)来对整个存储介质进行重新格式化，或是需要使用诸如BitLocker这样的加密工具。然而，重新格式化不是一个用户友好的过程，而诸如BitLocker这样的工具经常只提供卷级加密，而不是文件级加密。

[0021] 为解决这些问题，本公开内容的一些方面使得本机文件加密支持能够被集成到不提供此类支持的现有文件系统中，诸如FAT系的文件系统，而同时维持对这些文件系统的先前实现(即下级文件系统)的后向兼容性。

[0022] 下文描述的实施例是在某种FAT文件系统，特别是FAT32文件系统和exFAT文件系统的上下文中描述的。然而，要理解的是，本文描述的那些方面可以应用于采用目录项的目录来提供关于存储在存储介质上的文件的信息、但是目前不提供对文件级加密的本机支持的任何文件系统。因此，所要求保护的主题绝不限于FAT系的文件系统，且对FAT32、exFAT或任何其它FAT文件系统的讨论仅作为示例。

[0023] FAT32

作为进一步的背景，FAT32格式化的存储卷上的目录包括32字节目录项的线性表。FAT32上的每个文件都用一连串32字节的目录项来代表，诸如图2的示例中所示的用于具有文件名“The quick brown.fox”的文件的一连串目录项。如图所示，这一连串目录项包括短项，其以8.3字符格式存储缩短形式的文件名，以及关于该文件的额外信息，包括其创建时间、创建日期、最后访问日期、最后修改时间和最后修改日期。此外，该短项包括盘上的第一簇的标识，第一簇包含文件内容和文件大小。在文件目录中，该短项之后跟随着一个或多个用于捕获该文件的完整长文件名的长项。

[0024] 该短项的格式的进一步细节用以下的数据结构定义来表示：

```
typedef struct _PACKED_DIRENT {
    FAT8DOT3          FileName;          // 偏移 = 0
    UCHAR             Attributes;        // 偏移 = 11
    UCHAR             NtByte;            // 偏移 = 12
    UCHAR             CreationMSec;      // 偏移 = 13
    FAT_TIME_STAMP    CreationTime;      // 偏移 = 14
    FAT_DATE          LastAccessDate;    // 偏移 = 18
    union {                                                    。
        USHORT    ExtendedAttributes;    // 偏移 = 20
        USHORT    FirstClusterOfFileHi;  // 偏移 = 20
        };
    FAT_TIME_STAMP LastWriteTime;        // 偏移 = 22
    USHORT            FirstClusterOfFile;// 偏移 = 26
    ULONG32           FileSize;          // 偏移 = 28
} PACKED_DIRENT;
```

[0025] 该短项中没有字段被定义来存放任何形式的加密元数据。请注意，名为“NtByte”的字段在FAT文件系统中被定义为8比特的字段，其中的两个比特被使用，而其中的六个比特被定义为“保留的，忽略但保持该值”。因此，现有的FAT文件系统实现有效地忽略了该字段的保留位，但保持其值。在定义的两个比特中，一个比特定义短文件名的8个字符部分是否全部为小写，另一个比特定义短文件名的3个字符的扩展部分是否全部为小写。

[0026] 图3是图示用于在保持对文件系统的先前实现的后向兼容性的同时将本机加密支持集成到文件系统中的方法300的一个实施例的流程图。如图所示，该方法从步骤302开始，其中文件系统接收对于执行文件操作的请求。例如，该请求可以是对于创建文件、保存文件或对文件执行某其他操作的请求。在步骤304中，处理所述请求以确定其是否包括对该文件的内容进行加密的请求。这样的请求可以由与该请求相关联的一个或多个参数、标志、变元、变量等来指示，或者可以由请求本身的特定形式来指示。如果未指示对于加密的请求，则不执行图3的剩余步骤，并且文件系统按其常规方式处理该文件操作请求。然而，如果在步骤304中指示对文件进行加密的请求，则控制转到步骤306。请注意，尽管在图3中图示了步骤306至314，并且在下面以特定的顺序来描述，但该方法不限于所示步骤的特定顺序。

[0027] 在步骤306，给文件前加(prepend)文件头标。也就是说，文件头标将在文件的盘上表示中的文件内容之前写入。在一个实施例中，前加的头标的最小尺寸可以是存储介质的扇区大小，或者它可以是特定于介质的尺寸(例如，在闪存介质上为64k、对于硬盘驱动器为4k或512字节、或者是其整数倍)。这可以有助于性能(扇区大小可能不是特定于介质的性能(performant))。在一个实施例中，其中要在其上写入文件的存储介质的扇区大小为4KB，则这个头标可以是最小值为4KB，以便确保对文件主体的所有文件存取保持与扇区尺寸调准(align)。这可以消除对双缓冲非高速缓存的输入/输出(I / O)的需要并保持性能。在实施例中，通过从文件的开始处前加或去除整个簇，头标可以增长或缩小成超出初始最小大小

(例如,4KB)。头标的内容可以包括对解密该加密文件内容所需要的和在另外的情况下由文件系统进行管理所需要的任何加密元数据。例如,头标中的加密元数据可以包括用于加密文件的加密密钥、围绕文件的使用的策略、证书、初始化向量等。请注意,“前加”是逻辑前加。在文件盘区的列表中,文件头标数据将首先出现。在介质上,头标可以处在任何位置,即,是逻辑块地址(LBA)。

[0028] 在一个实施例中,如果在文件具有零文件长度时创建头标,则头标将默认大小为4KB。它可以通过以下方式来增长,即:将簇前加到文件的前面,并将头标重新定位回到该文件的分配的开头。如果文件在其中已经有内容,则头标创建可以默认为大小为一个簇,其最小大小为4KB,前加于该文件。如果头标需要增长,FAT文件系统将分配新的簇,将其插入到该文件中,然后重新定位头标内容以进行匹配。

[0029] 在一个实施例中,头标可以像单独的一段文件系统元数据那样被对待,并且上级FAT文件系统可能对可被向下发送到FAT驱动器的任何正常I / O请求分组(IRP)隐藏它。头标可以通过NtOfs API集合作为属性来显露,其中上级FAT实现可以提供最小的实现。

[0030] 在一个实施例中,头标可以按照*pfile*格式来格式化,*pfile*格式是由可从Microsoft Corporation(微软公司)获得的权限管理服务(RMS)创建的用于加密文件的标准头标格式。在其他实施例中,可以使用其他格式。例如,可以使用EFS文件系统的传统$EFS样式的加密头标。

[0031] 在步骤308,将用于该文件的加密元数据写入前加的头标。

[0032] 在步骤310,按照接收到的请求对文件的内容进行加密。在各种实施例中,可以利用各种各样加密算法中的任何一种来加密文件内容,加密算法是诸如高级加密标准(AES)、数据加密标准(DES)、Twofish、Serpent、Blowfish、CAST5、RC4、Triple DES等等。一旦被加密,经加密的内容便可以被写入到盘中,跟随在包含加密元数据的前加的头标之后。再次指出,加密文件的位置跟随在前加的头标之后是指由文件盘区描述的文件系统级的逻辑布局,而不是物理地址或LBA地址。在要由下级文件系统实现所使用的用于文件的正常文件目录项中,诸如图2所图示的短项,该项的首簇字段将标识前加的头标的开始,且文件大小将指示加密文件的完整大小,包括其头标和任何填充(如下讨论的)。

[0033] 在步骤312,在该实施例中,文件系统可以在用于该文件的文件目录项的保留(即未定义)字段中指示该文件被加密。在其中本方法在FAT32文件系统中被实现的一个实施例中,文件系统使用用于该文件的短目录项的NtByte字段的比特(bit)来指示该文件被加密。FAT32短目录项的NtByte字段具有保留的(即未定义的)六(6)个比特,且其可以用于此目的。在该实施例中,可以设置一个比特(例如,值=“1”)以指示“文件被加密”。此外,在一个实施例中,可以将第二比特设置为指示该前加的头标是标准大小(例如,4KB)。如果头标不是标准大小,则必须读取该头标以确定其大小。

[0034] 在一些实施例中,加密算法可能需要填充字节。例如,在一个实施例中,被采用来加密文件内容的加密算法可以是利用16字节块的AES链接块密码(AES-CBC)算法。这要求将所有文件大小四舍五入(round up)到最接近的16字节AES块,否则该文件的最后一段将无法恰当地解密。为了使下级文件系统实现能够读取文件的完整的最后一个块,盘上的文件长度被填充直到与下一个块调准——在AES-CBC的情况下,文件被填充直到与16字节调准,因此可能需要高达15个填充字节。因此,当在上级实现中打开或枚举文件时,上级实现需要

存储真实长度，或者从文件的填充长度来计算真实长度。在本实施例中，现有文件目录项的保留字段的额外比特可以用于存储被加到文件大小的填充字节的数量，或者替代地指示最后一个块中的非填充字节的数量。在大多数情况下，可以通过使用N个先前保留的比特来有效地高速缓存高达2^N-1个填充字节。在图3的方法被集成到FAT32文件系统中并且采用AES-CBC加密的实施例中，FAT短目录项的NtByte字段的四(4)个剩余的保留的(即未定义的)比特可以用于存储文件的最后16字节块的填充字节的数量(即N = 4)。为了使上级文件系统向用户呈现该文件，可以通过获取整个文件的文件大小、减去头标的大小、然后减去填充字节的数量来计算该文件的真实大小。在另一个实施例中，真实文件大小或文件大小增量也可以被留存在文件头标中，使得这个信息被保留用于冗余性或在制作了该文件的多个副本的情况下被保留。倘若NtByte字段被下级文件系统实现清除，则这种冗余性也可以有助于恢复。

[0035] 在采用利用16字节块的AES-CBC加密的实施例中，可以每512个字节设置初始化向量(IV)。在一个实施例中，IV被计算为以内容密钥作为加密密钥、利用AES来加密的文件中的512字节区的字节偏移；所得到的16字节块被用作IV。

[0036] 在步骤314中，加密文件的文件名被破坏(mangle)(即修改)，以帮助向上级文件系统指示该文件被加密，并使得加密文件能够对下级文件系统可见，以便如果期望的话则用户或应用可以采用用户模式工具来解密该文件。例如，在一个实施例中，在盘上，额外的扩展名(例如，“pfile”)可以被附加于文件名。例如，对于文件名为“The quick brown.fox”的文件，文件系统的下级实现将会看到名称“The quick brown.fox.pfile”。然后在被提供给正确的解密密钥时，与“.pfile”扩展名相关联的用户模式工具能够打开并解密任何这样的文件。在采用图3的方法的文件系统的上级实现上，该文件系统可以将要被视为本机加密文件的任何文件的“.pfile”扩展名隐藏起来，因此用户不需要担心使用工具来打开它们，且可以利用任何使用文件系统内的本机解密算法的应用来直接打开这些文件。

[0037] 对于额外的文件名扩展名(例如，“.pfile”)，在一些实施例中，上级文件系统可以执行如下的一个或多个额外的文件系统行为。首先，当创建一文件时，如果它有.pfile扩展名，则创建可能失败。其次，当创建文件时，文件系统可以检查name.doc和name.doc.pfile两者的存在。如果任一个已经存在，则例如由于已经存在的错误，创建可能失败。第三，当打开文件时，如果指定的文件以.pfile结尾并且没有标记为加密的，则打开可能失败。第四，当打开文件时，文件系统可以检查name.doc和name.doc.pfile两者，并打开任一个。如果它们都存在，则它可以打开它遇到的第一个。第五，当文件被重新命名时，文件系统可以检查目标目录中的name.doc和name.doc.pfile两者，并且如果存在任一名称，则可能使重新命名失败。第六，当枚举文件时，如果它有.pfile扩展名，则该扩展名可以被隐藏，除非在目录中已经遇到它的.pfile版本。

[0038] 一旦按照图3的方法将加密的文件存储到盘中，则当上级文件系统接收到文件存取操作(例如，“打开”文件的请求)时，文件系统可以依据在保留字段中设置的比特来确定该文件被加密，可以从文件名中剥除额外的扩展名(例如“.pfile”)，以及可以从文件头标中获得必要的加密元数据来解密该文件，以在文件系统内本机地供用户访问。此外，通过使用指示填充多少的那N个比特，可以有效地向用户显示经修改的文件大小，而无需读取前加的加密元数据头标。下级文件系统实现将忽略这些保留的比特，且只查看具有“.pfile”扩

展名的文件，从而维持后向兼容性。如果期望的话，则用户模式应用或工具(诸如IRM Reader)可以被用来解密文件。这要求盘上的文件大小——如在用于该文件的常规文件目录项中所反映的——必须反映加密的大小，加密的大小包括头标和对加密块大小的任何填充(例如，填充到AES块大小)。

[0039] 在一个实施例中，上级文件系统可以处置例如在目录的列表或需要文件大小的其他访问期间遇到文件的情况，其中文件名具有已添加的用来指示加密的扩展名(例如“pfile”)，但是其文件目录项的保留位(例如，FAT目录的短项中的NtByte字段)中的加密比特未被设置为指示该文件被加密。在检测到这种情况时，文件系统可以读取至少一些文件内容，以确定开头是否被证实为本文公开的类型的前加的头标。如果该文件内容被证实为前加的头标，则文件系统可以从前加的头标内容确定不带填充的真实文件大小，并且还可以进一步进行以下的任何组合：(a)将高速缓存结果保存在存储器中以提高对于该目录项的未来效率(b)如果存储介质是可写的，则更新该目录项中的加密比特和填充字节值，以使该文件系统的未来列表更高效，以及(c)隐藏额外的扩展名(例如“.pfile”)并报告未加密数据的“真实”大小。另一方面，如果该文件的内容未证实为加密文件，则文件系统可以如对于该文件在目录项(例如，FAT短项)中所表达的那样来报告文件名和文件大小。在其他实施例中，该过程可能不是作为文件系统的一部分实现的，而是作为另一系统组件实现的，该另一系统组件在命令文件系统设置加密比特之前执行对前加的头标的证实。换句话说，不是在枚举时由文件系统自动启用加密比特并证实前加的头标，而是该过程改而在用户例如通过使用用户模式进程(诸如外壳扩展、实用程序等)指令文件系统这样做时才按需执行。

[0040] 图4是图示用于在维持与文件系统的先前实现的后向兼容性的同时将本机加密支持集成到文件系统中的方法的另一实施例的流程图。在本实施例中，步骤402至410与图3的步骤302至310基本相同。当在步骤404确定该文件要被加密时，头标被前加于文件的盘上表示(步骤406)，加密元数据被写入头标(步骤408)，并且加密的文件内容被写入到盘上在该前加的头标的后面(步骤410)。如上文结合图3所讨论的，文件的内容可能需要被填充成与下一个加密块边界调准。但是在本实施例中，不是使用常规文件目录项的保留字段来指示文件被加密，而是在步骤412，在创建传统文件目录项之后(例如，在由图2的示例所图示的常规目录项之后)，在文件目录中创建额外的目录项。

[0041] 在步骤414，可以把真正的文件大小(例如，减去任何填充)和头标的大小直接存储在该额外的尾部目录项中。

[0042] 在步骤416，文件系统可以对文件的在先的常规目录项的至少一部分执行密码术(cryptographic)操作，并将结果存储在额外的尾部目录项中，以便为上级文件系统提供证实该新的尾部目录项与常规目录项相关联的手段。在一个实施例中，密码术操作包括对文件的在先的常规目录项计算校验和。

[0043] 在一个实施例中，额外的尾部目录项可以具有以下格式：

```
typedef struct _PACKED_DIRENT {
    UCHAR   UnusedMark;                 // 偏移 = 0
    UCHAR   Type;                       // 偏移 = 1
    UCHAR   Reserved[2];                // 偏移 = 2
    ULONG   AssociatedDirentsChecksum;  // 偏移 = 4
    ULONG   EfsHeaderSize;              // 偏移 = 8
    ULONG   FileSize;                   // 偏移 = 12
    UCHAR   Reserved[12]                // 偏移 = 16
    ULONG   Checksum;                   // 偏移 = 28
} PACKED_DIRENT;                        // sizeof = 32
typedef PACKED_DIRENT *PPACKED_DIRENT;
```

。

[0044] 该目录项的第一项UnusedMark标记其为未使用的FAT32目录项。如下一步所述的，该字段可以被设置为0xE5，表明该目录项是被删除的项。Type字段设置为0x1，可以将其定义为加密元数据。接下来的两个字节保留供将来使用。随后的字段AssociatedDirentsChecksum是用于文件的之前的一组常规FAT目录项的CRC32。接下来的两个ULONG直接存储加密文件的头标大小和真实文件大小。Reserved[12]字段也未被使用，以供将来的扩展。最后，Checksum字段是这一额外的目录项自身的CRC32。这是用来帮助证实这是一个真实的尾部目录项，而不是无用数据(junk)。请注意，文件的真实大小或真实大小与实际文件大小之间的差异也可以被存储在前加的头标本身中以用于冗余。倘若额外的尾部目录项被下级文件系统实现清除或删除，则这种冗余可以有助于恢复。

[0045] 在步骤418中，新的尾部目录项可以被标记为删除的目录项。例如，在以上所图示的实施例中，目录项的第一个字节可以设置为十六进制值“0xE5”。因为这个额外的尾部目录项被标记为未使用的或删除的，所以在下级文件系统实现中，它将被忽略。请注意，如果文件的常规文件目录项中的元数据被更改，则将使校验和无效。在这种情况下，上级文件系统实现可检测到这种无效，并重新生成包含其校验和的新的尾部目录项。这也是对FAT的上级实现的提示，以从文件的头标信息中重新计算头标大小和真实文件长度。

[0046] 步骤420与图3的步骤314基本相同，其中加密文件的文件名被破坏(即修改)，以帮助向上级文件系统指示该文件被加密，并且使得加密文件能够对下级文件系统可见，以便如果需要的话用户或应用可以采用用户模式工具来解密该文件。例如，在一个实施例中，在盘上，额外的扩展名(例如，“*.pfile*”)可以被附加于文件名。

[0047] 该实施例的优点在于，真实文件长度和头标长度可以高速缓存在目录本身中，而不是强制文件系统实现每一次都打开每个文件并从头标中读取这一信息。注意，使用短目录项的NtByte字段的4个保留位——如上文结合图3的实施例所论述的——也可以获得这一相同的功效。

[0048] 注意，本文所描述的且在图4中图示的方法可以被概括为：在目录中高速缓存任何类别的文件系统元数据信息，同时令没觉察到的(unaware)下级实现使高速缓存无效，而无需这个信息的结构的先验知识。

[0049] 前述的用于在现有文件系统(如FAT系的文件系统)中集成本机文件加密的方法使

得能够以后向兼容和执行的方式提供本机文件加密，以便可用于消费者文件加密、AppX应用包文件和应用数据(appdata)、以及像远程擦除和工作文件夹这样的企业场景。

[0050] 在实施例中，上级文件系统还可以将加密文件从一个目录(例如，目录A)移动到另一个目录(例如，目录B)。在一个实施例中，为了实现这样的移动，文件系统在目标位置(例如，目录B)中分配新的目录项。然后，文件系统从旧位置(例如，目录A)读取目录项的内容，并将内容写入目标位置的目录项。然后可以将旧目录项标记为未使用。在图3的实施例中，由于旧目录项的那些保留比特(例如NtByte字段)被复制，所以文件保有(retain)其加密性质。

[0051] 在下级文件系统中，文件从一个目录到另一个目录的移动也是可能的。在一个实施例中，这样的移动包括文件系统在目标位置中分配新的目录项，从旧位置的目录项中读取内容，将这些内容写入在目标位置中分配的目录项，然后把旧的目录项标记为未使用。再次地，由于NtByte被保留，但这些内容被定义为“保持但忽略”，所以该文件保有其加密性质。

[0052] 加密文件到非加密文件的转换也可以执行。在一个实施例中，这种能力可能不被文件系统直接支持，而是改而由加密服务在更高级别上执行，其创建新的未加密文件，然后把所有数据复制过来，且之后将新文件重新命名回旧名称。

[0053] 相反地，非加密文件到加密文件的转换也可以执行。再次地，在一个实施例中，这可能不被文件系统直接支持，而是改而由加密服务在更高级别上执行，其创建新的加密文件，然后把所有数据复制过来，且之后将新文件重新命名回旧名称。

[0054] 在一个实施例中，可以使用静态库模型把图3或图 4的方法集成到FAT32文件系统中，其中用于该方法的集成点被直接加到FAT文件系统中。在该实施例中，fastfat.sys二进制文件可以与静态库链接。分层和构建可以与NTFS的分层和构建相同；静态库可以被交叉仓库(cross depot)地发布，并被FAT文件系统使用(consume)。虽然用来实现所述集成的程序代码可以是处理器不可知的(agnostic)，但被所述实现调用来执行文件加密的密码库可以每平台地进行优化。

exFAT

[0055] 如上所述，本文所描述的且在图3和4中图示的方法也可以被具体化在exFAT文件系统实现中。作为进一步的背景，exFAT卷上的目录包括32字节目录项的线性表，类似于FAT32。exFAT卷上的文件用至少三个32字节目录项的序列来表示。第一个是FILE目录项，表示文件本身。第二个是STREAM目录项，表示文件中的数据，而第三个和随后的目录项是NAME目录项，表示文件的名称。

[0056] 在图3和4中被图示且在上文中被描述的方法可以以与上文相对于FAT32实现所描述的基本相同的方式在exFAT文件系统中被实现，除了(1)用来指示“文件被加密”的比特(即，标志)的目录项位置，(2)标识文件的最后16字节块的填充字节数的信息的目录项位置，以及(3)指定前加的文件头标长度的长度的目录项中的信息的供应和位置之外。

[0057] 在exFAT文件系统实现的一个实施例中，exFAT FILE目录项的Reserved0字段的先前保留的比特(例如，比特0x8000)被用于以本文所述的方式指示“文件被加密”。以下描述了exFAT FILE目录项的格式，包括新定义的“文件被加密”比特：

```
typedef struct_FPP_FILE_RECORD {
```

```
    //
    // 类型将具有值 'Dirent' (0x85)
    //
    FPP_RECORD_HEADER Type;
    //
    // 跟随这一项的次要项的计数. 对于'FileName' (0xC0)
    // 必须有至少一个。
    //
    UCHAR Secondary Entries;
    //
    // 对主要和所有次要项计算的校验和。
    //
    USHORT Checksum;
    //
    // 标准文件属性。
    //
    USHORT Attributes;
    //
    // 保留以用于将来的扩展。
    //
    USHORT Reserved0; (这个项的比特0x8000被定义以意指:该文件被加密;以前这是保留的比特)
    //
    // 当该文件被创建和写入时以及在该文件被创建和写入处
    //记录下面的八个字段。
    //
    FAT_TIME_STAMP CreationTime;
    FAT_TIME_STAMP LastWriteTime;
    FAT_TIME_STAMP LastAccessTime;
    UCHAR CreationTimeMsec;
    UCHAR LastWriteTimeMsec;
    FAT_TIME_ZONE CreationTimeZone;
    FAT_TIME_ZONE LastWriteTimeZone;
    FAT_TIME_ZONE LastAccessTimeZone;
    UCHAR Reserved1 [7];
} FPP_FILE_RECORD, *PFPP_FILE_RECORD;
```

在其他实施例中,可以使用不同的先前保留的比特来指示“文件被加密”。

[0058] 进一步地按照在exFAT文件系统实现中的实施例,在exFAT STREAM目录项格式的先前保留的字段中定义一新的EfsTailByteCount字段,用于存储文件的最后16字节块的填

充字节数，并且在exFAT STREAM目录项的另一个先前保留的字段中定义一个EfsHeaderLength字段，用于存储前加的文件头标的长度的指示。以下描述按照一个实施例的、exFAT STREAM目录项的经修改的格式：

```
typedef struct_FPP_STREAM_RECORD {
  //
  // 类型将具有值'Stream' (0xC0)
  //
  FPP_RECORD_HEADER Type;
  UCHAR AllocationFlags;
  //
  // EfsTailByteCount。这是在最后的AES密码块中
  // 的EOF之后的字节的数量，即，该文件的最后16字节块
  // 的填充字节的数量。对于AES这通常是在0到15
  // 之间的数。未来的密码算法可能有更大的块尺寸。
  //
  UCHAR EfsTailByteCount; (以前是保留的字段)
  //
  // 文件名中字符的计数，横跨必须紧跟在这个记录之后的
  // 那些随后的FPP_NAME_RECORD 记录。
  //
  UCHAR CharCount;
  //
  // 这个字段是全部大写的名称的散列。当检查文件名
  //冲突时，这允许快得多的比较。
  //
  USHORT UpcasedNameHash;
  //
  // EfsHeaderLength。这是如上所述的和图3中图示的、
  // 就4096字节块而言，前加于文件的开始的头标的长度。
  // 4096 字节 = 1 个块。
  // 头标的最大尺寸是256 KB，它是64 个块。
  //
  UCHAR EfsHeaderLength; (以前是保留的字段)
  UCHAR Reserved2;
  LONGLONG ValidDataLength;
  ULONG Reserved3;
  //
  // 定义文件数据居于(lives)何处。
  //
```

```
    FAT_ENTRY FirstClusterOfFile;
    LONGLONG FileSize;
} FPP_STREAM_RECORD, *PFPP_STREAM_RECORD;
```

请注意，在本实施例中，STREAM记录中的EfsTailByteCount字段可以由一个字节(即8个比特)组成，而不是由在FAT32实现中定义的4个比特组成，这允许容纳高达256字节的块大小。另外请注意，在本实施例中，没有对exFAT NAME目录项格式进行改变。且如上所述，除了上面说到的对exFAT FILE和STREAM目录项格式的改变之外，在上文被描述且在图3和4中被图示的方法在exFAT文件系统实现中可以以基本相同的方式操作。

[0059] 本文构想的实施例包括以下：

1．一种计算设备，其包括处理单元、存储器和在所述处理单元上执行的文件系统，当所述文件系统在所述处理单元上执行时，其完成以下操作，包括：

响应于指示对文件进行加密的请求的文件操作，给该文件前加一个头标；

加密所述文件的内容，并将加密的内容写入所述文件中所述头标之后；

将与该加密文件相关的加密元数据存储在所述头标内；

在用于所述文件的文件目录项的保留字段中指示该文件被加密；和

将额外的文件名扩展名附加于该加密文件的文件名，进一步指示该文件被加密。

[0060] 2．实施例1所述的计算设备，其中所述文件系统设置所述保留字段的单个比特来指示所述文件被加密。

[0061] 3．实施例1或2中任一项所述的计算设备，其中所述前加的头标具有一个初始的预定大小，其可以由所述文件系统进行扩展，所述文件系统用所述保留字段的第二比特来指示该前加的头标是否超过所述初始的预定大小。

[0062] 4．实施例1至3中任一项所述的计算设备，其中所述加密操作包括使用基于固定块大小操作的加密算法来对文件的内容进行加密，并且其中所述文件系统还执行以下操作，包括：

把该文件的内容填充成与最后的加密块的块大小调准；和

把填充量的指示存储在用于所述文件的对于文件目录项的保留字段中。

[0063] 5．实施例4所述的计算设备，其中所述文件系统还在前加的头标中存储以下的至少一项：(a)该加密文件的真实大小的指示或 (b)该填充量的指示。

[0064] 6．实施例1至5中任一项所述的计算设备，其中所述文件系统还执行以下操作，包括：

检测如下的文件，即：其具有的文件名带有指示该文件是加密文件的额外的已附加的扩展名、但其中该文件目录的保留字段并未指示该文件被加密：

读取至少一些文件内容，以确定开头是否被证实为其中存储有加密元数据的前加的头标；和

如果所述至少一些文件内容被证实为前加的头标，则进一步执行以下的一项或多项：

(a) 将高速缓存的结果留在存储器中，以提高对于这个目录项的未来效率；

(b) 如果其上存储该文件的存储介质是可写的，则更新用于所述文件的目录项的保留字段，以指示该文件被加密；和

(c) 向用户隐藏该额外的扩展名，并报告未加密数据的真实大小。

[0065] 7．实施例1至6中任一项所述的计算设备，其中所述文件系统还执行以下操作，包括：

通过以下方式将该加密文件从第一目录移动到第二目录：

在第二目录中分配一个新的目录项；

从第一目录中读取所述目录项的内容；

将所述内容写入第二目录的所述新目录项；和

将第一目录的所述目录项标记为未使用。

[0066] 8．一种计算设备，其包括处理单元、存储器和在所述处理单元上执行的文件系统，当所述文件系统在所述处理单元上执行时，其完成以下操作，包括：

响应于指示对文件进行加密的请求的文件操作，给该文件前加一个头标；

加密所述文件的内容，并将加密的内容写入所述文件中；

将与该加密文件相关的加密元数据存储在所述头标内；

为所述文件创建额外的文件目录项，其跟随在文件目录中的用于该文件的其他目录项之后；

将关于该文件的真实大小的信息存储在所述额外的文件目录项中；

对用于该文件的其他文件目录项的至少一部分执行密码术操作，并将密码术操作的结果存储在所述额外的文件目录项中；和

将所述额外的文件目录项标记为被删除的或未使用的文件目录项。

[0067] 9．实施例8所述的计算设备，所述文件系统还将额外的文件名扩展名附加到所述文件的文件名。

[0068] 10．实施例8或9中任一项所述的计算设备，其中所述文件系统还将所述前加的头标的大小存储在所述额外的文件目录项中。

[0069] 11．实施例8至10中任一项所述的计算设备，其中对用于所述文件的其他文件目录项的至少一部分执行的所述密码术操作包括计算校验和。

[0070] 12．实施例8至11中的任一项所述的计算设备，其中所述加密操作包括使用基于固定块大小操作的加密算法来加密所述文件的内容，并且所述文件系统把所述文件的内容填充成与最后的加密块的块大小调准，被存储在额外目录项中的关于所述文件的真实大小的信息反映所述文件在没有填充的情况下的大小。

[0071] 13．实施例12所述的计算设备，其中所述文件系统还在前加的头标中存储以下的至少一项：(a)加密文件的真实大小的指示或 (b)填充量的指示。

[0072] 14．一种在计算设备的文件系统中提供本机文件加密支持的方法，包括：

响应于指示对文件进行加密的请求的文件操作，给该文件前加一个头标；

加密所述文件的内容，并将加密的内容写入所述文件中所述头标之后；

将与该加密文件相关的加密元数据存储在所述头标内；

在用于所述文件的文件目录项的保留字段中指示该文件被加密；和

将额外的文件名扩展名附加于所述加密文件的文件名，进一步指示该文件被加密。

[0073] 15．实施例14所述的方法，所述指示包括设置所述保留字段的单个比特来指示所述文件被加密。

[0074] 16．实施例14或15中的任一项所述的方法，其中所述前加的头标具有一个初始的

预定大小，其可以由所述文件系统进行扩展，所述方法还包括用所述保留字段的第二比特来指示前加的头标是否超过初始的预定大小。

[0075] 17．实施例14至16中的任一项所述的方法，其中所述加密操作包括使用基于固定块大小操作的加密算法来对所述文件的内容进行加密，并且所述方法还包括：

把所述文件的内容填充成与最后的加密块的块大小调准；和

把填充量的指示存储在用于所述文件的对于文件目录项的保留字段中。

[0076] 18．实施例17所述的方法，还包括在前加的头标中存储以下的至少一项：(a) 对于加密文件的真实大小的指示或 (b) 对于填充量的指示。

[0077] 19．实施例14至18中的任一项所述的方法，所述文件目录项包括FAT32文件系统的短目录项，并且所述保留字段包括所述短目录项的NtByte字段。

[0078] 20．实施例14至19中的任一项所述的方法，所述文件目录项包括exFAT文件系统的FILE目录项，并且所述保留字段包括exFAT FILE目录项的Reserved0字段。

[0079] 本文描述的那些方面的例图旨在提供对各个方面的结构的一般性理解。这些例图并不打算用作对于使用本文所描述的结构或方法的装置和系统的所有元件和特征的完整描述。在考查本公开内容后，许多其它方面对于本领域技术人员来说可能是明显的。可以从本公开内容利用并导出其它方面，使得可以在不脱离本公开内容的范围的情况下做出结构和逻辑上的替换和改变。因此，本公开内容和附图被认为是说明性的而不是限制性的。

[0080] 结合本文所公开的那些方面来描述的各种说明性逻辑块、配置、模块和方法步骤或指令可以被实现为电子硬件或计算机软件。各种说明性的组件、块、配置、模块或步骤已经就其功能性被大体上描述。这样的功能性是被实现为硬件还是软件取决于特定的应用和施加在整个系统上的设计约束。所描述的功能性可以针对每个特定的应用以变化的方式来实现，但是这样的实现决策不应当被解释为导致偏离本公开内容的范围。

[0081] 结合本文所公开的那些方面来描述的各种说明性逻辑块、配置、模块和方法步骤或指令，或者它们的某些方面或某些部分可以以存储在计算机可读存储介质上的计算机可执行指令(即，程序代码)的形式来体现，其中所述指令在由诸如计算设备这样的机器执行时，执行和/或实现本文所描述的系统、方法和过程。具体地，可以以这样的计算机可执行指令的形式来实现上述的任何步骤、操作或功能。计算机可读存储介质包括以用于存储信息的任何非暂时的(即，有形的或物理的)方法或技术实现的易失性和非易失性、可拆卸和不可拆卸介质，但是这样的计算机可读存储介质不包括信号。计算机可读存储介质包括但不限于RAM、ROM、EEPROM、闪速存储器或其他存储器技术、CD-ROM、数字通用盘(DVD)或其他光盘存储装置、盒式磁带、磁带、磁盘存储装置或其他磁存储设备、或者可用于存储所需信息并且可由计算机访问的任何其它有形的或物理的介质。

[0082] 尽管已经用特定于结构特征和/或动作的语言描述了所述主题，但是要理解，在所附权利要求中限定的主题不一定限于上述的特定特征或动作。相反，所述特定特征和动作是作为实现权利要求的示例被公开的，并且其他等同的特征和动作打算归入权利要求的范围内。

[0083] 提供了对那些方面的描述以使得能够进行或使用所述方面。对这些方面的各种修改将容易是明显的，并且在不脱离本公开内容的范围的情况下，本文定义的一般性原理可以应用于其它方面。因此，本公开内容不打算限于本文所示出的那些方面，而是被给予与由

以下权利要求限定的原理和新颖特征一致的最广可能范围。

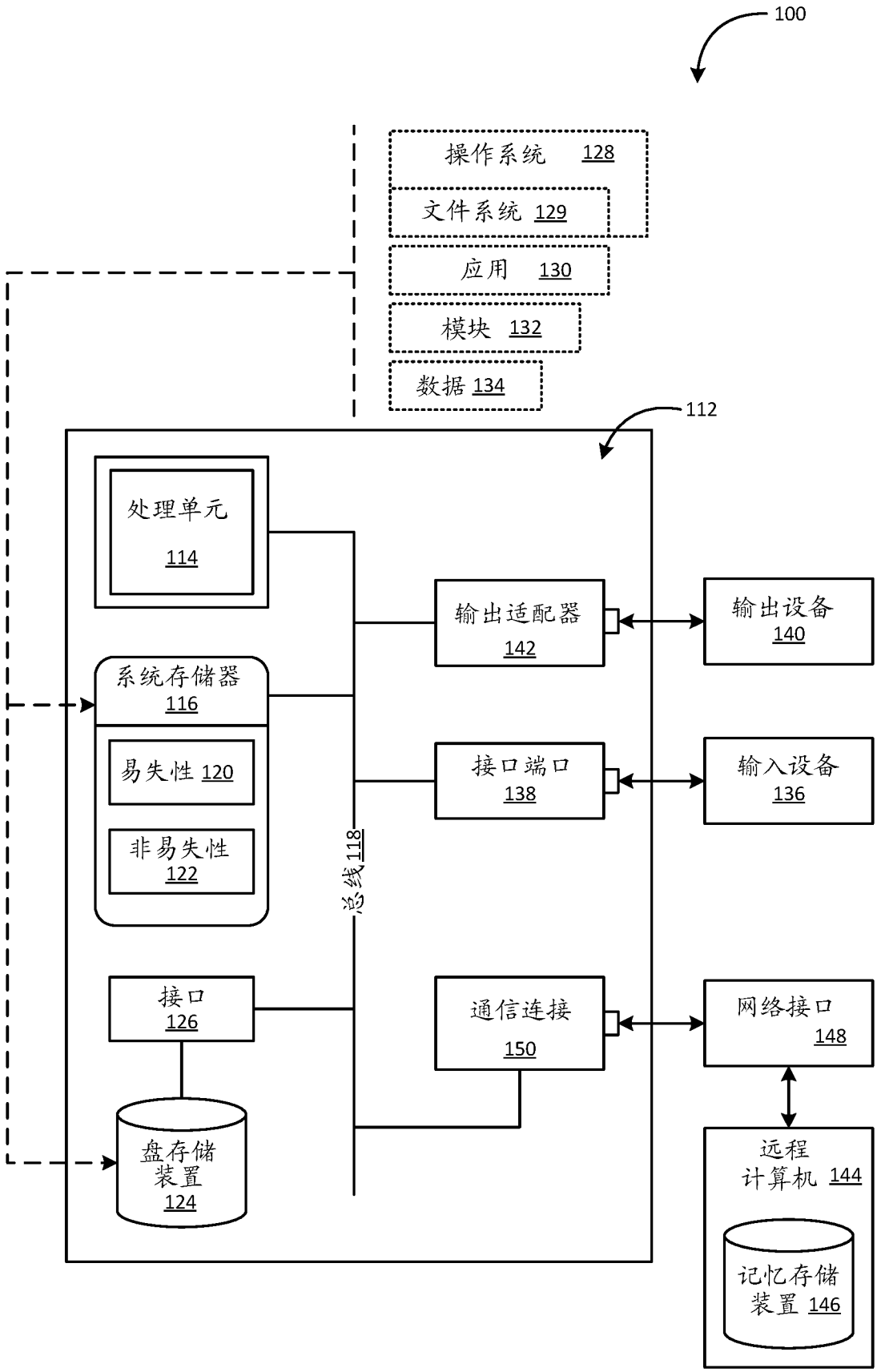

100
操作系统 128
文件系统 129
应用 130
模块 132
数据 134
112
处理单元 114
系统存储器 116
易失性 120
非易失性 122
总线 118
接口 126
盘存储装置 124
输出适配器 142
输出设备 140
接口端口 138
输入设备 136
通信连接 150
网络接口 148
远程计算机 144
记忆存储装置 146

图 1

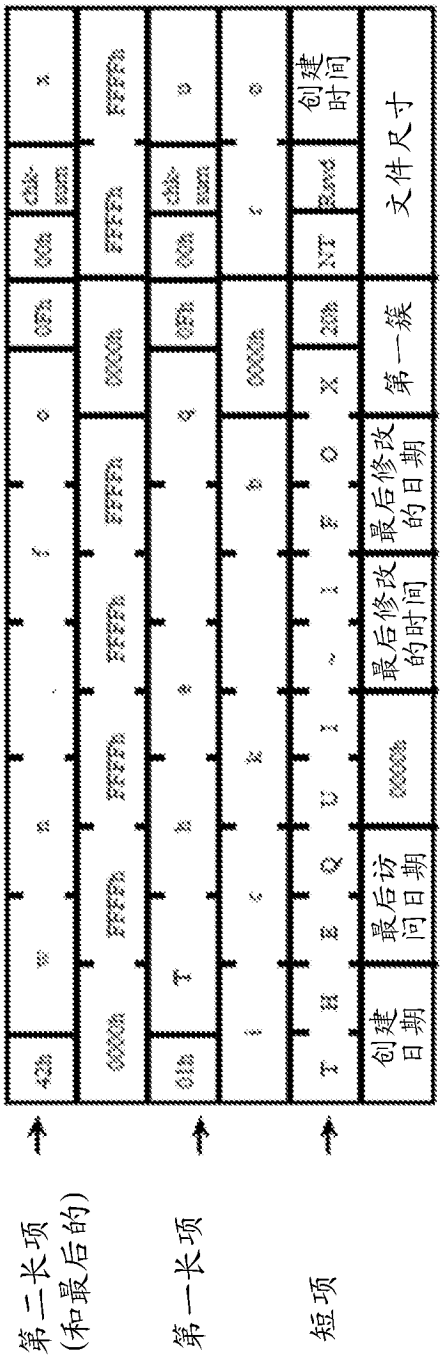


图 2

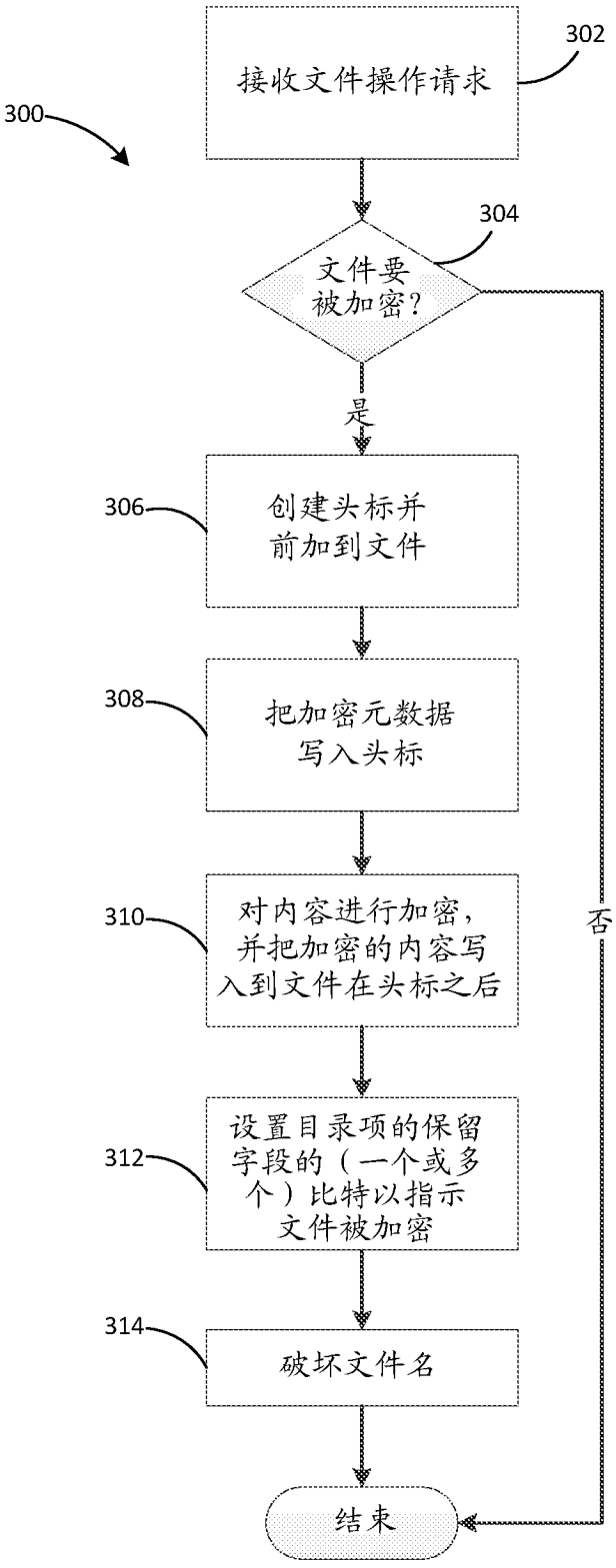

300
接收文件操作请求
302
文件要被加密?
304
是
否
306
创建头标并前加到文件
308
把加密元数据写入头标
310
对内容进行加密,并把加密的内容写入到文件在头标之后
312
设置目录项的保留字段的(一个或多个)比特以指示文件被加密
314
破坏文件名
结束

图 3

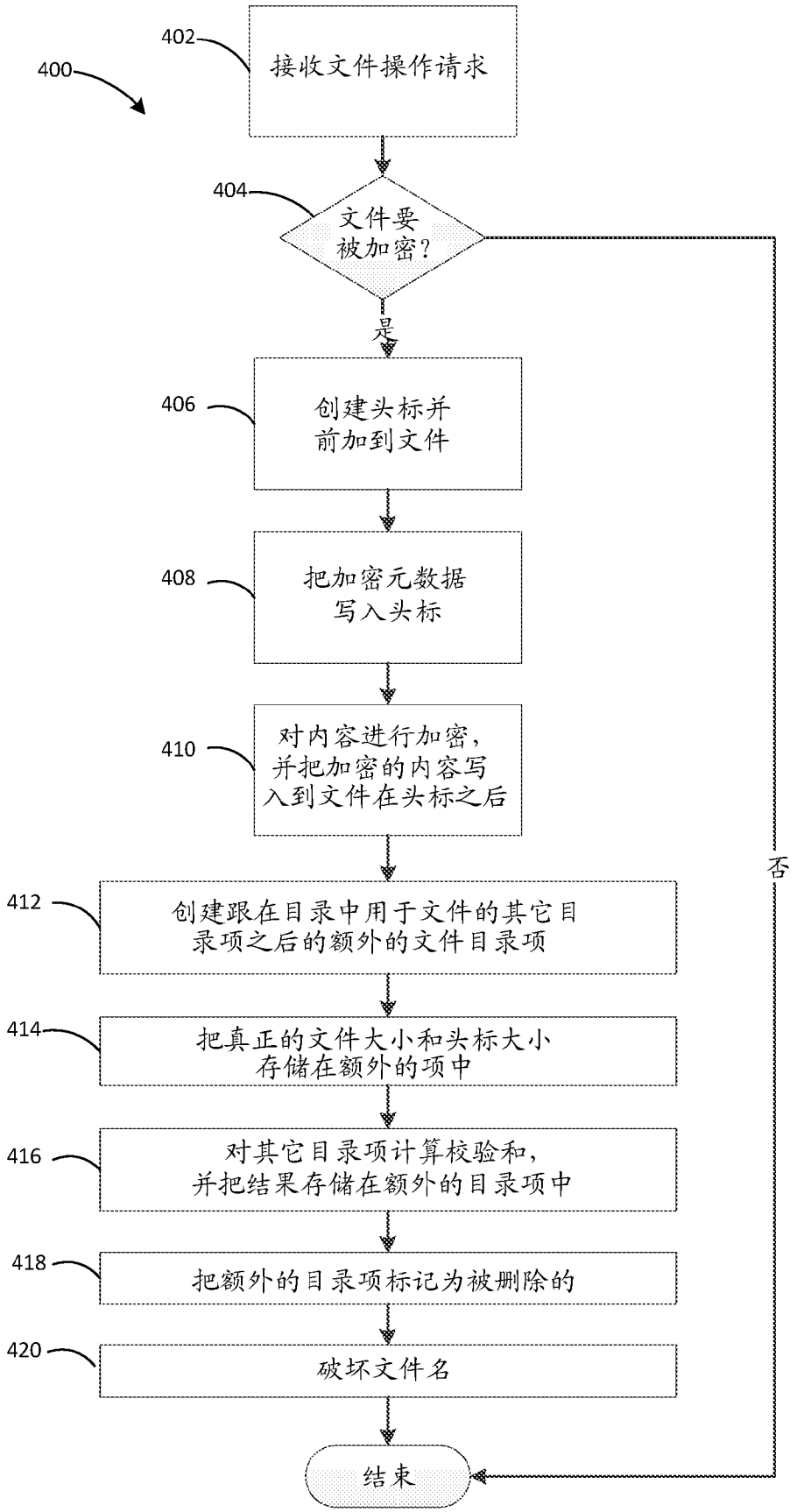

400
402
接收文件操作请求
404
文件要被加密？
是
否
406
创建头标并前加到文件
408
把加密元数据写入头标
410
对内容进行加密，并把加密的内容写入到文件在头标之后
412
创建跟在目录中用于文件的其它目录项之后的额外的文件目录项
414
把真正的文件大小和头标大小存储在额外的项中
416
对其它目录项计算校验和，并把结果存储在额外的目录项中
418
把额外的目录项标记为被删除的
420
破坏文件名
结束

图 4