北京大学学报(自然科学版) 第 54 卷 第 1 期 2018 年 1 月
Acta Scientiarum Naturalium Universitatis Pekinensis, Vol. 54, No. 1 (Jan. 2018)
doi: 10.13209/j.0479-8023.2017.088

基于安全目录的文件保护机制

沈熳婷 俞银燕† 汤帜 崔晓瑜

北京大学计算机科学技术研究所, 北京 100080; † 通信作者, E-mail: yuyinyan@pku.edu.cn

摘要 针对现有的文件保护方法侧重于保护单个设备上文件信息的特点, 面向跨设备使用和保护文件信息的需求, 提出一种高效的基于安全目录的文件保护机制。该机制可动态地检测安全目录中的用户行为, 实时保护文件信息, 支持文件多设备安全共享。实验结果表明, 基于安全目录的文件保护机制具有良好的可用性、响应速率和存储性能。

关键词 安全目录; 文件保护机制; 多设备; 加密; 访问控制

中图分类号 TP399

An Efficient Safe Directory Based File Protection Mechanism

SHEN Manting, YU Yinyan†, TANG Zhi, CUI Xiaoyu

Institute of Computer Science & Technology, Peking University, Beijing 100080;
† Corresponding author, E-mail: yuyinyan@pku.edu.cn

Abstract Existing file protection approaches and the applications are suitable for the users who own one single device but not the users with multiple devices. A novel mechanism is proposed that provides an efficient and fast cryptographic-based protection approach for files in an appointed directory called safe directory. The highlights are its ability to protect sensitive files automatically in a short time and the support for secure file sharing across devices. All operations in the safe directory are monitored in real time and the files in the directory are protected automatically. Experimental results show that proposed mechanism performs well in performance.

Key words safe directory; file protection mechanism; multiple devices; encryption; access control

随着信息技术的快速发展, 文件数字化变得越来越普及, 文件的信息安全问题也随之而来。大型公司和机构往往会购买专用的文件保护系统, 对公司的机密文件进行管理和保护。该类文件保护系统因其昂贵的价格和部署需求等方面的原因, 不适用于个人信息的保护, 因此需要设计一种适用于个人用户的文件保护机制。

文件保护有很多方法, 包括使用数字版权保护(digital rights management, DRM)技术[1]对数字内容进行保护和追踪, 以及使用安全软件对敏感文件进行保护。DRM 可以控制文件内容的使用和传播[1−2], 并且能够控制其只能被已授权的用户以一定的规则在规定期限内使用。由于个人用户和小型组织的文件量较少, 使用具有内容服务器和许可证服务器的 DRM 体系结构来设计文件保护机制显得冗余, 使用本地文件保护机制则更合适。

控制用户对敏感文件的访问也是一种有效的文件保护方式。访问控制可以通过预先规定用户对于文件所能执行的操作权限, 控制用户对敏感文件的访问和使用。根据控制条件, 访问控制分为基于角色的访问控制(role-based access control, RBAC)[3]、基于用户行为的访问控制以及使用控制(usage control, UCON)[4−5]等。基于角色的访问控制将权限授予角色, 同一用户可以继承不同的角色来获取不同的权限, 同一角色也可由多个用户继承[6], 用户通过建立会话来访问资源。

863 计划(2012AA013102)资助
收稿日期: 2016−11−03; 修回日期: 2016−12−13; 网络出版日期: 2017−05−19

本文要解决的是个人用户文件的安全问题，无需建立角色和用户的多对多关系，因此 RBAC 不适用于解决该问题。UCON 作为下一代访问控制的概念模型，在诸多方面对传统的访问控制进行了扩展和延伸，定义了授权、义务和条件 3 个要素，并提出连续性和可变性两个重要属性[7]。UCON 提高了数字作品的安全性，适应使用环境的可变性。但是，其连续性的实现要求在用户使用文件的过程中，不间断地或重复地检查用户是否有资格进行或继续对文件的访问，而在个人用户文件保护应用中无需重复检查权限。

近年来，各种针对计算机文件系统的数据加密保护技术不断发展完善，其中，加密文件系统(encrypting file system, EFS)以其较高的易用性和安全性被广泛关注[8]。EFS基于操作系统的用户账号和权限管理，与文件系统集成在一起，对用户完全透明。EFS 为每个登录操作系统的用户生成特有的公私钥对，用以生成文件密钥以及解密加密后的文件内容，而这些操作对用户都是透明的，一旦用户成功登录操作系统，就可以对文件进行读写操作。由于EFS使用RSA等成熟的加密算法[9]，安全性也相当高。但是，EFS认为设备间相互独立，即使对不同的设备使用相同的管理员密码，也不能在设备间进行机密文件的安全共享。特别地，只有NTFS格式的windows分区才可以使用EFS加密技术。并且，将敏感文件从具有加密属性的文件夹复制到非加密文件夹中，文件会自动解密，意味着将敏感文件在不同设备中传输会使文件被自动解密和暴露。因此EFS不能满足用户跨设备管理以及保护敏感文件的需求。

Zhao 等[10]使用虚拟文件系统存储敏感文件，认为普通的文件保护机制提供的访问控制可能被攻击者绕过，因此提出一种分离的虚拟机存储技术。用户对敏感文件的请求都只能通过 guest 虚拟机发出，由 admin 虚拟机处理。这样分离的文件保护方式使攻击者无法绕过文件保护机制，而必须遵守系统的访问控制规则，尤其适用于已经存在被感染威胁的操作系统。但是，对于用户数量较少且固定的情况，这样的系统结构响应速率低，时延较高。

除上述文件保护系统和方法外，各种文件保护工具也越来越多地开发出来并投入使用，这些文件保护工具默认当前安装工具的设备为唯一的受保护信息的载体。这种以设备而非用户为单位管理和保护文件的方法不支持用户在多个设备上共享和保护文件。随着科技的发展和用户需求的提高，平板电脑、手机等移动设备也成为用户使用敏感信息的平台，现有的各种单机文件保护工具已经不能满足需求。对拥有多个设备的用户而言，更需要一种能够以用户为单位，适用于同一用户的多个设备的文件保护机制。

当前，通过云平台提供的云存储服务存储和共享数字文件已经逐步为人们所熟悉，如用户可以通过 Office365 或 OneDrive 等实现信息的及时更新和共享，数据文件的跨设备共享和同步非常方便，但此类应用要求用户完全信任提供服务的服务器。同时，因为云计算具有开放特性和资源共享特性，针对云计算的新型攻击方式已经出现，如基于公共物理机的旁通道攻击等。因此，有必要探索一种新的文件保护机制，从源头保护敏感文件，减少对云服务器的高度可信依赖，使用户在享受云存储服务进行信息文件的存储、传输和共享的同时，避免因文件泄漏而导致的隐私泄漏。

本文设计一种高效的基于安全目录(一个存储敏感文件并提供自动保护功能的目录)的文件保护机制(safe directory based file protection mechanism, SDFPM)，为用户提供安全透明的敏感文件保护功能，并能自动探测用户行为，实时自动保护文件。SDFPM 机制还能够为用户提供高效便携的跨设备文件保护支持。为了克服现存文件保护软件只能保护特定格式文件的缺陷，SDFPM 定义一种新的安全文件结构，使得任意格式的文件都能够按照该结构被封装和保护。为了保证文件的安全性，任何在安全目录中的文件都能够被实时动态地保护，防止由于用户遗忘对文件进行加密而带来的文件信息泄露。本文的主要贡献在于，定义一种统一的安全文件结构来扩展文件的保护范围，为持有多设备的用户提供一种跨设备的文件保护方式，使得用户能够透明便利地操作被保护的文件。

1　SDFPM 框架

SDFPM 的主要目标是设计一种能够有效保护用户敏感文件，支持跨设备保护的文件保护机制，框架如图 1 所示，主要由设备管理模块、文件监视器和文件保护模块组成。

随着用户拥有设备的增加，面向单设备的文件保护工具对于同时拥有多个设备的用户而言已经不

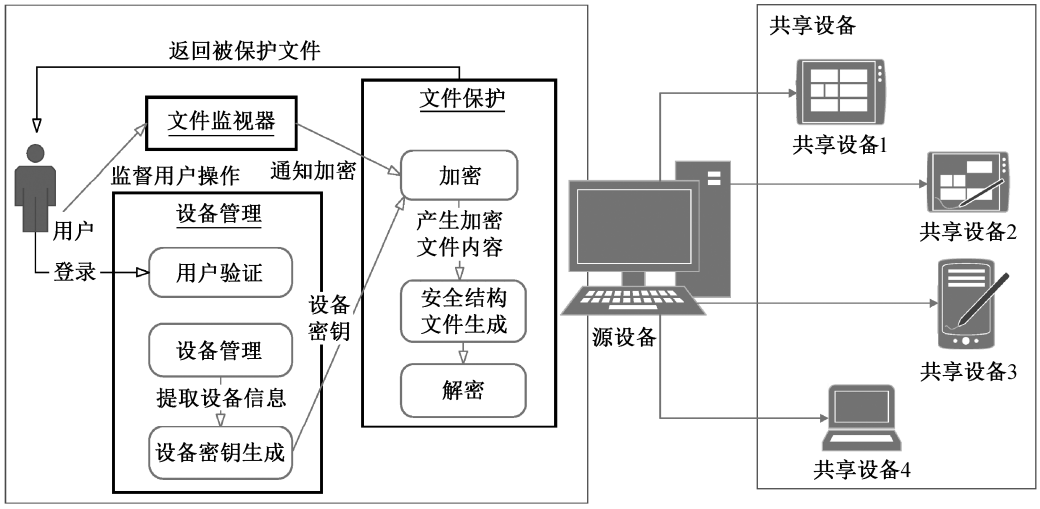

图 1 SDFPM 框架
Fig. 1 Framework of SDFPM

够用。现有的文件保护软件一旦在某个设备上成功安装，就能且仅能对所安装设备上的文件进行保护。对于拥有多个设备的用户，更需要一种能够进行跨设备文件管理和保护的机制，使用户在持有的不同设备中都能对自己的文件进行操作。SDFPM 的设备管理模块解决了现有软件无法跨设备文件保护的问题，能够用于实现安全目录的访问控制以及对同一用户的多个设备的管理(包括设备的添加和删除等)。

当一个文件被攻击者截获，如果信息以明文的形式存储在文件中，那么攻击者可以直接获得文件中的信息。因此，理想的文件保护机制需要最小化文件中的信息以明文形式存在的时间。现有的文件保护机制是仅当用户明确指出需要保护的文件时，才对文件提供保护，用户的遗忘和疏忽会直接导致明文信息的泄露，因此这样的保护方式对用户管理以及文件较多的使用场景不够理想。SDFPM 设计了文件监视器来尽可能地减小敏感文件信息以明文形式存在的时间。文件监视器对进入目录的任何文件进行实时监视，并将监测结果提交文件保护模块，用于实时监控用户对安全目录中文件的行为，对于用户的操作能够及时检测并处理，是文件保护模块的指示器。

目前，基于加密的文件保护工具通常直接使用用户输入的密码生成文件加密密钥。出于记忆考虑，用户往往选择有意义的密码(如生日和姓名等)，这样的加密密钥无法抵御穷举攻击。并且，由于存在被保护文件的密钥由用户自己定义的情况，用户出于方便记忆等原因，文件的密钥之间会存在一定的相互关联，一旦某个文件的密钥被破解，系统中其他被保护文件都会变得不安全。因此，为了对文件提供更安全的保护，需要设计更有效的密钥生成算法。文件保护模块用于接收文件监视器传递的信号，为每一个文件生成不同的加密密钥，且对用户透明，无须用户记忆，以实现对敏感文件的加密封装保护以及对被保护文件的解密。

2 SDFPM 文件保护机制

2.1 设备管理

设备管理模块分为用户验证、设备管理和设备密钥生成 3 个部分。用户验证用于实现对请求访问安全目录用户的访问控制，设备管理用于管理一个合法用户的所有注册设备，包括设备添加和删除。考虑到用户拥有设备的可变性，定义用户创建被 SDFPM 保护的敏感文件的设备为该文件的源设备，源设备能够将被保护的敏感文件分享给其他设备，被分享被保护文件的用户设备(如笔记本电脑和平板电脑)为共享设备。源设备与共享设备拥有形式相同的安全目录，由于安全目录是被保护文件的载体，因而不同设备间敏感文件仅在设备的安全目录之间传递。

2.1.1 用户注册与验证

设备管理模块根据用户请求创建安全目录后，初始化设备信息表，设备信息表中包含当前设备的

设备信息$<\text{"local"}, \text{Dev}_{\text{dig}}, \text{Dev}_{\text{info}}>$, Dev_{dig}为设备验证码，Dev_{info}为设备硬件信息，$\text{Dev}_{\text{dig}} = H(\text{Dev}_{\text{info}})$，$H(\cdot)$为单向哈希函数。设备之间进行敏感文件的共享 时，首先验证文件共享后的存储载体是否为合法的安全目录，当且仅当安全目录的设备验证码验证通过时，才允许用户对文件进行后续操作。

用户在设备端使用安全目录保护数字文件前，需要先输入账号密码进行注册。注册用户的用户信息以 ticket 的形式进行存储，以保证用户的账号信息和密码信息不被窃取。用户注册完成之后，其 ticket 由式(1)生成，并使用系统秘密密钥 K_s 加密为 $E_{K_s}(\text{ticket})$ 保存在设备中。一旦用户修改密码，他的 ticket 就会重新生成和存储，并删除旧的 ticket。

$$\text{ticket}_u = H(\text{password}_u \oplus \text{ID}), \tag{1}$$

其中，password_u是用户密码; ID是该用户的用户名，也是该用户在机制中的唯一标识；$H(\cdot)$代表单向哈希函数，确保即使 ticket 泄露，攻击者也无法通过ticket来反向计算出用户密码等信息。

用户在设备端操作安全目录的文件前，需要先输入账号密码进行身份认证。机制不直接验证用户的账号密码，目的是在不揭露用户密码的情况下验证用户身份的合法性。用户认证过程如图 2 所示，当用户输入用户名密码进行登录时，设备管理模块会自动生成临时 ticket，同时解密获取该用户对应于该安全目录的正确 ticket，并进行对比，当且仅当两者完全一致时，用户才被认证为合法，并允许进入安全目录进行文件操作。

2.1.2　设备添加与删除

本文参考基于遍历加密的多设备内容共享与版权保护方法[11]中的设备管理机制，给出设备管理方法。

首先定义设备信息表。设备信息表由用户所有授权设备的设备信息及其 MAC 码组成，用以实现受保护文件在各授权设备上的安全共享。授权设备的设备信息是一个三元组，包括设备标签、设备验证码和设备硬件信息。设备标签在授权设备添加时由用户录入，各个授权设备的设备标签各不相同，设备验证码为设备硬件信息的单向 Hash 值。设备信息表可以表示为

$$\text{Dlist} = (\{< \text{Dev}^i_{\text{tag}}, \text{Dev}^i_{\text{dig}}, \text{Dev}^i_{\text{info}} > \mid i = 1, ..., n\}, \text{Adev}_{\text{mac}}),$$

n 为当前授权设备总数，$< \text{Dev}^i_{\text{tag}}, \text{Dev}^i_{\text{dig}}, \text{Dev}^i_{\text{info}} >$为第 i 个授权设备的设备信息。

设备验证码 $\text{Dev}^i_{\text{dig}}$ 和设备硬件信息 $\text{Dev}^i_{\text{info}}$ 之间的关系可表示为

$$\text{Dev}^i_{\text{dig}} = H(\text{Dev}^i_{\text{info}}),$$

其中，$H(\cdot)$代表单向哈希函数，用于检验设备信息的正确性。

设备信息表的验证码和授权设备信息三元组的关系为

$$\text{Adev}_{\text{mac}} = \text{HMAC}_{K_s}(\{< \text{Dev}^i_{\text{tag}}, \text{Dev}^i_{\text{dig}}, \text{Dev}^i_{\text{info}} > \mid i = 1, \cdots, n\}),$$

其中，K_s为秘密系统密钥。

设备信息表与当前设备的设备硬件信息 Dev_{info}

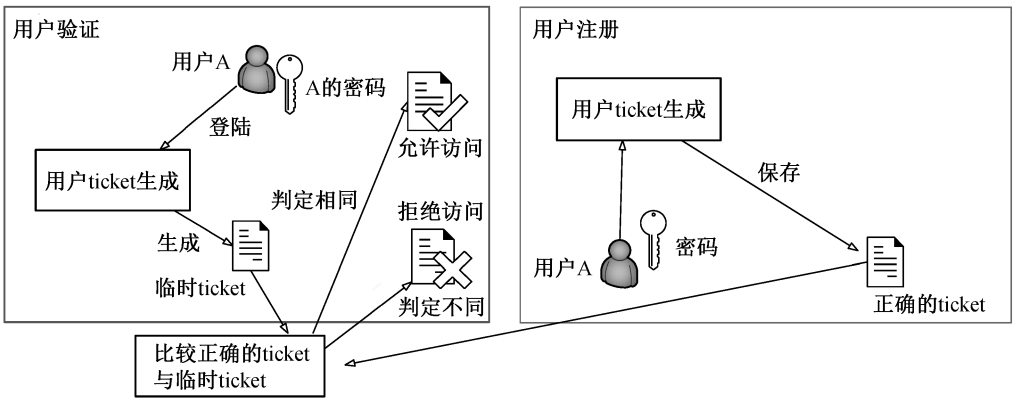

图 2　用户认证过程

Fig. 2　User verification process

绑定，以加密的形式存储在设备上:

$$E_{G^a(\{\mathrm{Dev}_{\mathrm{info}},K_s\})}(\mathrm{Dlist}),$$

其中，$G^a(\cdot)$ 为对称密钥生成函数，与对称加密算法 a 相关。

接着，给出设备添加和设备删除过程。

1) 设备添加过程。

步骤 1 设备管理模块提取设备的硬件信息 $\mathrm{Dev}_{\mathrm{info}}$，使用公式 $\mathrm{Dev}_{\mathrm{dig}}=H(\mathrm{Dev}_{\mathrm{info}})$ 生成设备信息的验证码，构建二元组 $<\mathrm{Dev}_{\mathrm{dig}},\mathrm{Dev}_{\mathrm{info}}>$，使用系统密钥 K_s 和用户输入的授权码 A_c 进行加密: $E_{G^a(\{K_s,A_c\})}(<\mathrm{Dev}_{\mathrm{dig}},\mathrm{Dev}_{\mathrm{info}}>)$，然后发送给源设备。为增强安全性，也可由用户线下安全传送(如拷贝等)。

步骤 2 源设备的设备管理模块接收待添加设备的信息后，要求用户输入授权码，并根据系统密钥和授权码解密获取该设备的设备验证码和设备硬件信息 $<\mathrm{Dev}_{\mathrm{dig}},\mathrm{Dev}_{\mathrm{info}}>$。检验 $\mathrm{Dev}_{\mathrm{dig}}=H(\mathrm{Dev}_{\mathrm{info}})$ 是否成立。若不成立，则说明待添加设备信息有误，结束设备添加过程。

步骤 3 若检验满足 $\mathrm{Dev}_{\mathrm{dig}}=H(\mathrm{Dev}_{\mathrm{info}})$，说明该设备信息无误。源设备的设备管理模块获取当前设备的设备硬件信息和系统密钥，解密设备信息表 $D_{G^a(\{\mathrm{Dev}_{\mathrm{info}},K_s\})}(\mathrm{Dlist})$，并获取设备信息表信息

$$\mathrm{Dlist}=(\{<\mathrm{Dev}^i_{\mathrm{tag}},\mathrm{Dev}^i_{\mathrm{dig}},\mathrm{Dev}^i_{\mathrm{info}}>$$

$$|\,i=1,...,n\},\mathrm{Adev}_{\mathrm{mac}})。$$

检验 $\mathrm{Adev}_{\mathrm{mac}}=\mathrm{HMAC}_{K_s}(\{<\mathrm{Dev}^i_{\mathrm{tag}},\mathrm{Dev}^i_{\mathrm{dig}},\mathrm{Dev}^i_{\mathrm{info}}>|\,i=1,...,n\})$ 是否成立，若不成立，则说明设备信息表信息有误，结束设备添加过程。

步骤 4 将设备信息中的设备验证码与接收到的 $\mathrm{Dev}_{\mathrm{dig}}$ 逐一匹配进行查重。如果设备表中已经存在与 $\mathrm{Dev}_{\mathrm{dig}}$ 相同的记录，说明该设备已被添加，结束设备添加过程。

步骤 5 设备管理模块要求用户输入接收到的待添加设备的唯一设备标签，若该标签已在设备表中被使用，要求用户更换标签，否则，使用用户输入的标签和 $<\mathrm{Dev}_{\mathrm{dig}},\mathrm{Dev}_{\mathrm{info}}>$ 生成设备信息三元组 $<\mathrm{Dev}^{n+1}_{\mathrm{tag}},\mathrm{Dev}^{n+1}_{\mathrm{dig}},\mathrm{Dev}^{n+1}_{\mathrm{info}}>$。

步骤 6 源设备将新产生的设备信息添加到设备信息表中，并使用秘密系统密钥重新生成设备表的 MAC 码，并完成整个设备表的更新:

$$\mathrm{Dlist}^*=(\{<\mathrm{Dev}^i_{\mathrm{tag}},\mathrm{Dev}^i_{\mathrm{dig}},\mathrm{Dev}^i_{\mathrm{info}}>$$

$$|\,i=1,...,n,n+1\},\mathrm{Adev}^*_{\mathrm{mac}}),$$

$$\mathrm{Adev}^*_{\mathrm{mac}}=\mathrm{HMAC}_{K_s}(\{<\mathrm{Dev}^i_{\mathrm{tag}},\mathrm{Dev}^i_{\mathrm{dig}},\mathrm{Dev}^i_{\mathrm{info}}>$$

$$|\,i=1,...,n,n+1\})。$$

使用源设备的设备硬件信息和系统密钥对更新后的设备信息表进行加密并存储。

2) 设备删除。

删除一个设备，需要从设备信息表删除该设备所对应的信息。对删除的设备，需要根据其唯一设备标签删除该设备所对应的记录。删除后的设备将因无法还原出文件密钥解密文件内容而成为无效设备。具体过程如下。

步骤 1 设备管理模块接收到进行设备删除的请求，生成源设备的设备硬件信息，解密设备信息表，得到设备表中记录的设备信息和设备表消息验证码，并利用

$$\mathrm{Adev}_{\mathrm{mac}}=\mathrm{HMAC}_{K_s}(\{<\mathrm{Dev}^i_{\mathrm{tag}},\mathrm{Dev}^i_{\mathrm{dig}},\mathrm{Dev}^i_{\mathrm{info}}>$$

$$|\,i=1,...,n\})$$

计算设备表的 MAC 码，当且仅当新计算出的 MAC 码和存储于设备表中的设备信息表的 MAC 码一致时，继续删除过程，否则终止。

步骤 2 设备管理模块获取所有授权设备的设备标签并呈现给用户。

步骤 3 用户选择待删除设备的设备标签，并提交删除请求。

步骤 4 设备管理模块根据用户指定的设备标签找到相应的设备记录，删除该记录。重新根据现有的设备信息表内容生成消息认证码后更新设备表，使用源设备的设备硬件信息和系统密钥加密设备信息表并存储。

3) 设备密钥的生成。

设备密钥用于保护被保护文件的文件密钥，该文件密钥用于加密文件的内容数据。设备密钥在文件加密和解密过程中动态生成，对不同的被保护文件，用于加密被保护文件的文件密钥的设备密钥不同。设备密钥由设备硬件信息和对应文件的盐生成，设备 D 的设备密钥的生成可以表示为

$$K_D=G^a(\{\mathrm{Dev}_{\mathrm{info}},R,K_s\}),$$

其中, $G^a(\cdot)$为对称密钥生成函数, 与对称加密算法 a 相关, $\mathrm{Dev}_{\mathrm{info}}$ 为设备 D 的设备硬件信息, R 为文件加密过程中为被保护文件生成的盐, K_s 为秘密系统密钥。

2.2　文件监视器

文件监视器作用于文件保护前, 作为用户和受保护文件的交互桥梁。一旦用户在安全目录中进行影响文件安全性的操作, 文件监视器能够实时探测到这些操作, 并通知文件保护模块对文件进行保护。用户无须对文件手动进行加密操作, 所有的加密封装过程都会在用户创建或修改文件后自动完成。

1) 系统服务原理。

以使用 Windows 操作系统的 PC 源设备为例, 实现文件监视器的方式是使用book service (挂钩系统服务)将本机制监控相关的代码挂接在系统文件服务函数上。系统服务被 Windows 操作系统封装在底层, 其二进制代码主要存放在 ntoskrnl.exe 中, 用户态下的程序必须通过wrapper function接口来访问系统服务。Wrapper function 又称为本地API, 以动态链接库的形式提供, 存放在ntdll.dll中。编写程序调用Win32 API, 使用wrapper调用系统服务, 对编写的代码进行参数有效性检查, 并转换为unicode形式, 发出int 2e指令。处理器根据中断号在系统中断描述符表(IDD)中找到中断处理函数的起始地址, 将参数从用户态堆栈拷贝到核心态堆栈, 然后根据服务 ID 号查询系统服务分配表, 获得服务函数的地址并开始执行。监控相关的代码从挂接到调用系统服务的主要流程如图 3 所示。

2) 文件监视器实现方式。

本机制通过将编写的监视代码挂接在系统文件服务函数上, 实现对安全目录中文件操作的事实监控。修改 SSDT, 将系统服务函数指针改为本机制的监视代码, 同时保存原有函数地址。挂接的监视代码主要包括驱动程序和监视交互程序。驱动程序用于监控安全目录中文件的变化, 发送监控信息给监视交互程序, 监视交互程序则过滤收到的文件改变信息, 对于有效文件操作(影响文件安全性的操作), 通知文件保护模块做出响应。

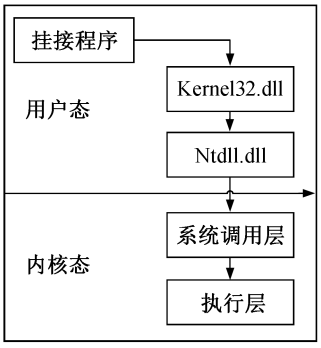

图 3　调用系统服务主要流程

Fig. 3　Main procedure of calling book service

2.3　文件保护

文件保护模块实现受保护文件的加密封装与解封解密, SDFPM 保护的文件按照定义的安全文件结构进行封装。

2.3.1　安全文件结构

目前流行的加密软件(如超级文件夹加密大师)仅对特定格式的文件进行保护。然而, 小企业中需要保护的文件的信息是多样的, 文件格式也无法预知, 因此本文设计一种安全文件结构, 确保任意格式的文件都能有效地被加密封装和保护。

安全文件结构(safe file structure, SFS)包括两部分: 文件头和文件内容。文件头包括以下内容。

1) 文件总长度: 一个整数, 指示文件总长度。

2) 文件头长度: 一个整数, 指示文件头长度。

3) 盐: 一个随机数 R, 用于防止文本猜测攻击[3], 同时也用于恢复文件密钥。

4) 文件密钥密文: 用于加密文件内容的文件内容密钥 CEK 的密文, 使用各授权设备的设备密钥加密后, 存储在文件头中, 用于文件解密。

如图 4 所示, 文件密钥密文由密文总长度、密文信息总项数、文件密钥密文信息和密钥认证码组成。密文总长度记录文件密钥密文的总长度, 密文信息总项数记录与各授权设备绑定的密文信息总数(与当前总授权设备个数一致), 密钥认证码由公式 $\mathrm{HMAC}_{K_s}(\mathrm{CEK})$ 产生, 用于验证文件解密过程中设备恢复的文件密钥的正确性。密文信息 i 包括与某个授权设备绑定的文件密钥密文 $E_{G^a(\mathrm{Dev}_{\mathrm{info}}^i, R, K_s\})}(\mathrm{CEK})$ 及该设备的设备验证码 $\mathrm{Dev}_{\mathrm{dig}}^i = H(\mathrm{Dev}_{\mathrm{info}}^i)$, $\mathrm{Dev}_{\mathrm{info}}^i$ 为该设备的设备硬件信息。

5) 初始文件的消息认证码(MAC): 原始文件内容的 MAC 码, 可采用基于 SHA256 的 HMAC, 密钥使用文件密钥, 用于对解密后的文件内容进行完整性验证。

6) 初始文件长度: 原始的文件长度, 用于文件

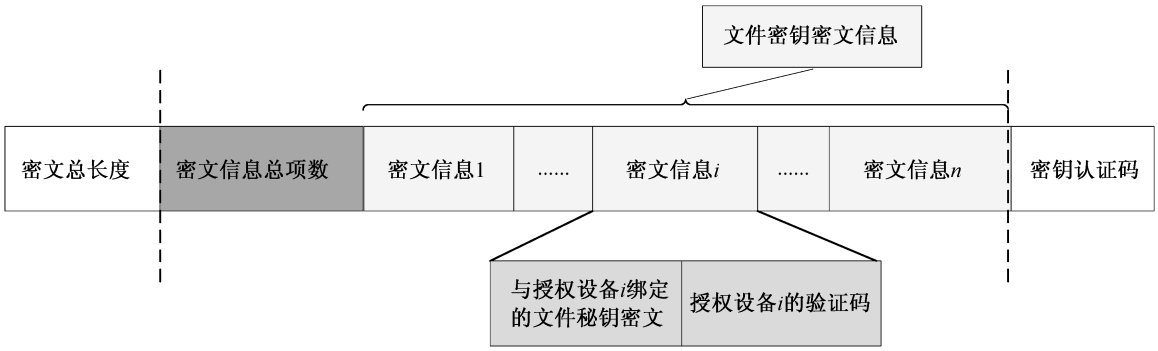

图 4 文件密钥密文项结构

Fig. 4 Detail structure of encrypted content key information

恢复。

7) 文件头的消息摘要: 计算上述所有信息的消息摘要, 用于后续解密前的文件头的完整性验证。

文件密文内容不同于文件头, 安全文件结构中的文件内容是用文件内容密钥对称加密的原始文件内容的密文。只有能够恢复出正确的文件密钥的设备才能查看敏感文件的原始内容。

2.3.2 文件加密保护与解密

用户在源设备中创建文件后, 源设备会对文件进行加密保护, 并封装为 SFS 文件, 存储在安全目录中。用户将源设备安全目录中的 SFS 文件传递到合法的共享设备的安全目录中, 共享设备便能够对该文件进行解密查看等操作。不同设备的安全目录创建方式相同, 可通过源设备的设备信息表记录的设备信息摘要加以联系与区分。当用户创建或更新一个文件后, 文件保护模块将对文件进行加密封装, 具体过程如下。

步骤 1 文件保护模块接收到文件监视器发送的文件保护请求后, 记录待保护文件的文件长度, 为待保护文件随机生成文件内容密钥 CEK 和盐 R, 使用内容密钥加密文件内容, 生成文件内容密文, 并产生该文件内容的消息认证码, 然后向设备管理模块请求授权设备密钥。

步骤 2 设备管理模块获取本设备的设备硬件信息, 解密设备信息表, 检验设备信息表的完整性后, 获取所有相关授权设备的信息

$$\{< \mathrm{Dev}^i_{\mathrm{tag}}, \mathrm{Dev}^i_{\mathrm{dig}}, \mathrm{Dev}^i_{\mathrm{info}} > | \, i = 1, ..., n\} \text{。}$$

根据设备密钥生成方法产生各授权设备的设备密钥

$$K^i_D = G^a(\{\mathrm{Dev}^i_{\mathrm{info}}, R, K_{\mathrm{s}}\}),\ i = 1, ..., n \text{。}$$

步骤 3 设备管理模块将产生的设备密钥和对应的设备验证码返回给文件保护模块。

步骤 4 文件保护模块使用每一个设备密钥, 遍历加密 CEK 生成与该授权设备绑定的文件密钥, 使用系统密钥产生密钥认证码, 根据设备密钥数量、与各设备绑定的文件密钥密文和对应的设备验证码等信息, 生成受保护文件的文件密钥密文项。

步骤 5 利用前 4 步产生的信息, 根据 SFS 安全文件结构, 生成 SFS 文件头及其消息摘要, 并进一步生成受保护的 SFS 文件;

步骤 6 (可选) 如果需要将被保护文件在设备间共享, 则需要用户将 SFS 文件传递到共享设备的安全目录中。传递到另一设备的 SFS 依旧保持加密状态, 如果传递到安全目录外, 则会因为设备信息摘要验证不通过而无法对文件进行解密等操作。

如果支持共享设备对受保护文件的编辑功能, 则需要源设备将授权设备信息表使用目标共享设备硬件信息和系统密钥生成的密钥加密后, 共享给该共享设备; 若仅允许共享设备查看受保护的文件, 则不需要。

一个 SFS 文件可能是存储的当前设备生成的, 也可能是由源设备生成后传递到该共享设备的。无论该设备是源设备还是共享设备, 对于同一个 SFS 文件的解密方式完全一致。当用户需要在一个合法设备(包括产生加密文件的源设备和被分享文件的共享设备)查看被保护的敏感文件内容时, 需进行文件解密, 具体过程如下。

步骤 1 文件保护模块接收到文件监视器发送的文件查看请求后, 获取文件头的内容, 生成当前文件头内容的消息摘要, 并与存储在文件头中的文件头摘要进行对比, 验证文件头中存储信息的完整

性，保证文件头中存储的相关信息没有被篡改。

步骤 2　文件保护模块读取受保护文件头中记录的盐 R，发送给设备管理模块。设备管理模块获取本设备的硬件信息 $\mathrm{Dev}_{\mathrm{info}}$，使用系统密钥 K_s 和盐 R 产生该设备的设备密钥 $K_D = G^a(\{\mathrm{Dev}_{\mathrm{info}}, R, K_s\})$，同时生成 $\mathrm{Dev}_{\mathrm{info}}$ 的单向 Hash 值，一并发送给文件保护模块。

步骤 3　文件保护模块读取文件头中的文件密钥密文，遍历所有文件密钥密文信息项 $<\mathrm{ECK}^i, \mathrm{HD}^i>$，读取每一项的设备验证码 HD^i，并与设备管理模块生成的本机设备硬件信息的单向Hash值进行逐一对比。若某一项编号为 i的密文信息的设备验证码与当前设备的硬件信息的单向Hash值完全相同，则说明该项密文信息对应于该设备，若没有一项能够匹配，说明该设备非法或已经被删除，终止文件查看过程。

步骤 4　对于匹配上的文件密钥密文信息项，使用步骤 2 生成的设备密钥解密该密文信息项中的文件密钥密文 $\mathrm{CEK} = D_{K_D}(\mathrm{ECK}^i)$，获取文件内容密钥 CEK，检验 HMAC_{K_s} (CEK) 与文件密钥密文中存储的密钥认证码是否相等，不相等则解密失败，终止文件查看过程。

步骤 5　使用文件内容密钥 CEK 对 SFS 的文件密文内容进行解密，得到原始文件内容。进一步生成该原始文件内容的消息认证码，并与记录在文件头中的初始文件的消息认证码进行比对，验证初始文件内容的完整性。若相等，则展现文件内容，否则终止文件查看过程。

这些过程对用户完全透明，不影响用户对文件的使用。

3　实验和分析

3.1　相关工作

目前流行的文件保护工具有 Folder Protection①、超级文件夹加密大师(GFEM)和 Bitlocker②等。

Folder Protection 是使用率较高的快捷加密软件，通过使用用户输入的密码作为文件密钥来加密用户文件。对于任意被保护的文件，Folder Protection 都要求用户首先选择该文件的路径，输入密码后才开始对文件进行加密保护。如果用户由于疏忽而忘记对文件进行保护，一旦攻击者窃取了该文件，文件信息会完全暴露。这样的保护方式依赖于用户的行为，对敏感文件的保护不是实时自动的。

超级文件夹加密大师是目前较流行的多功能文件加密软件，保护文件和文件夹的方式同样使用对称加密算法进行加密。然而，对于某些格式的文件③，文件夹加密大师无法对其进行保护，在加密这些格式的文件时，会自动跳过而不进行保护。理想的文件保护方式是能够独立于被保护的文件的格式，对所有格式的文件都进行加密保护。

Bitlocker 是微软开发的数据加密工具，通过加密个人电脑操作系统数据卷上的信息，达到文件保护的目的。实验表明，使用 Bitlocker 会在一定程度上影响电脑性能，一台 16 G 内存的笔记本电脑运行 Bitlocker，分别加密 16，64 和 512 MB 的分区块时，写速率分别下降 18.6%, 8.0%和 27.2%。本文提出的文件保护机制的加密单位为单个文件，可以大大提高文件保护速率。4 种文件保护工具的性能分析如表 1 所示。

除 SDFPM 外，其余 3 种文件加密工具虽然都为用户提供文件保护服务，但却无法满足多设备的需求。由于所提供的文件保护功能均基于设备，对于同一个用户的多个设备，文件保护工具将他们视为完全独立的对象，不支持文件共享。SDFPM 在保护文件的前提下，为用户拥有的多个设备提供认证和文件共享的功能，适用于多设备的使用环境。

3.2　实验及性能分析

实验机器使用 2.5 GHz 主频率的 CPU，所有实验均在 Windows 操作系统上运行。实验采用的加密算法为 AES-256 加密算法(CBC 模式), Hash 算法采用 SHA-256 算法，MAC 算法采用基于 SHA-256 的 HMAC 算法。

3.2.1　时间性能

本机制的多设备文件保护方案不指定用户使用设备的数量，用户可以根据需要添加和删除设备。为了体现机制的时间有效性，首先使用单个设备上的 SDFPM 与其他 3 种文件保护工具进行文件保护实验，并测试各工具的文件保护时间。

① http://www.kakasoft.com/folder-protect/how-to-lock-directory.html

② https://msprotect.microsoft.com/home/Campaign?campaignId= 63b37527-d5fa-40e5-9f1c-cbb8c6df097b

③ 如使用 7-ZIP 文件压缩软件产生的 7z 格式的文件。

表 1 相关工作对比

Table 1 Comparision of related work

文件保护工具	是否自动保护	是否影响设备读写速率	是否支持所有格式	是否直接使用用户密码	是否支持多设备
SDFPM	是	否	是	否	是
Folder Protection	否	否	否	是	否
GFEM	否	否	否	是	否
Bitlocker	是	是	是	否	否

首先分别计算在不同文件大小情况下，SDFPM 生成一个受保护的安全文件(从用户保存关闭到文件生成)和恢复被保护文件为原始文件(从用户点击打开到文件解密恢复)的时间。为了更直观地分析时间性能，我们定义性能变量 IR_{open} 和 IR_{save}。

IR_{open} 指在使用 SDFPM 打开文件时，执行解密恢复的时间与用户点击打开到文件打开完毕的时间的比例:

$$IR_{open}=\frac{time_{restoration}}{time_{open}+time_{restoration}}。\tag{2}$$

IR_{save} 指在使用 SDFPM 保存新建或修改后的文件时，执行加密封装的时间与用户点击保存到加密文件生成和保存的时间的比例:

$$IR_{save}=\frac{time_{generation}}{time_{save}+time_{generation}}。\tag{3}$$

实验挑选具有普遍性的 txt 和 doc 文件，给出不同大小的文件使用和不使用 SDFPM 机制进行文件打开和关闭所需要的时间，同时给出 IR_{open} 和 IR_{save}，如表 2 所示。

从表 2 可以看出，使用 SDFPM 保护文件的加密时间远少于文件打开和保存关闭的固有时间，因此 SDFPM 机制对文件操作时间的影响非常小。我们使用其他 3 种文件保护工具对敏感文件进行保护，并测试各个工具在文件保护时所需要的加密时间，将其与 SDFPM 进行对比(其中，Bitlocker 加密存储对应大小文件的分区)，结果如表 3 所示。

从表 3 可以看出，由于 Bitlocker 在加密被保护文件的同时也加密了许多额外数据，因此时间性能较差。Folder Protection 和 GFEM 在加密时间性能上与 SDFPM 相当，但 SDFPM 在确保良好的时间性能的基础上，具有更广泛的适用性。

为了评估设备的增加对 SDFPM 时间性能产生的影响，我们分别使用 SDFPM 在不同设备数量情况下对 txt 格式文件进行保护，结果如表 4 所示。

表 2 不同大小文件的操作时间

Table 2 Operating time for different file sizes

文件类型	操作		操作时间/s		
			10 KB	1 MB	10 MB
txt	打开	N	0.13	0.22	9.57
		Y	0.14 (7.1%)	0.25 (12.0%)	9.77 (2.0%)
	保存关闭	N	0.15	0.35	19.57
		Y	0.16 (6.3%)	0.38 (7.9%)	19.75 (0.9%)
doc	打开	N	0.98	1.01	5.22
		Y	0.99 (1.0%)	1.04 (2.9%)	5.42 (3.7%)
	保存关闭	N	0.51	2.75	15.51
		Y	0.52 (1.9%)	2.78 (1.0%)	15.69 (1.1%)

说明: 括号内数值为影响比率 IR; “N”和“Y”分别代表不使用和使用 SDFPM 机制进行文件保护时打开/关闭一个文件需要的时间。

表 3 不同工具加密时间对比

Table 3 Comparison of different tools

文件保护工具	加密执行时间/s		
	10 KB	1 MB	10 MB
SDFPM	0.01	0.03	0.18
Folder Protection	0.07	0.72	1.03
GFEM	0.13	0.18	0.54
Bitlocker	0.38	5.31	8.32

从表 4 可知，使用 SDFPM 进行多设备共享文件保护的耗时随设备数量的增加而增加。时间的差异主要体现在文件关闭前生成设备密钥并遍历加密文件密钥的时间。但是，即使设备个数较多，SDFPM 所需时间也较少。

3.2.2 空间性能

为了评估 SDFPM 所需要的存储空间，我们通过实验计算 SDFPM 保护不同大小的文件(txt 格式)所需要的额外存储空间，实验结果如表 5 所示。从表 5 可以看出，使用 SDFPM 保护不同大小的文件

表 4　多设备加密时间对比
Table 4　Muti-device encryption time

授权设备个数	操作	总时间/s		
		10 KB	1 MB	10 MB
1	打开	0.140	0.250	9.770
	关闭	0.161	0.380	19.752
3	打开	0.143	0.253	9.800
	保存关闭	0.166	0.433	19.870
6	打开	0.162	0.270	9.972
	保存关闭	0.282	0.385	21.750

表 5　保护不同大小的文件所需额外存储空间
Table 5　Extra space of SDFPM for different file sizes

初始文件大小	SFS 文件大小	额外空间	额外空间所占比例/%
10 KB	10.13 KB	135 B	1.350
1 MB	1000.12 KB	125 B	0.013
10 MB	10.00 MB	123 B	0.001

所带来的额外空间需求非常小，不依赖于原始文件大小。

表 6 给出使用不同文件保护工具时，对于不同大小的 txt 格式被保护文件，被保护之后生成的文件的总存储空间。

如表 6 所示，Bitlocker 产生的额外空间是最大的，接近于原始文件大小的 2.5 倍。SDFPM 的额外存储空间最小，且最稳定。当文件大小显著增加时，额外的存储空间尤其会成为存储空间不大的移动存储设备(如手机、平板)的负担，我们更希望保护文件所产生的额外空间可以预估且相对小和稳定，在保护文件的同时，不影响设备其他方面的使用。

因此，我们计算了每个文件保护工具的文件信息占比(初始文件大小/被保护的文件大小，即待保护的初始文件大小与被保护后的文件大小的比例)，并进行比较，结果如图 5 所示。可以看出，SDFPM 的文件信息占比相对较高且十分稳定。

表 6　存储空间对比
Table 6　Storage comparison

文件保护工具	受保护文件占用的存储		
	10 KB	1 MB	10 MB
SDFPM	10.13 KB	1.00 MB	10.00 MB
Folder Protection	1286 KB	2.24 MB	11.24 MB
GFEM	13 KB	1.12 MB	10.64 MB
Bitlocker	26.1 KB	2.56 MB	24.77 MB

3.3　安全性分析

设备信息的安全性。初始源设备保存所有授权设备的设备信息(设备表)，使用该设备的设备硬件信息和秘密系统密钥生成的密钥进行加密存储。在设备硬件信息的提取方法保密的情况下，设备信息表是安全的，秘密系统密钥的使用也增强了设备信息表的安全性。设备信息表需要传递给其他授权设备时，仅能由源设备解密并使用目标授权设备的设备硬件信息和系统密钥产生的密钥进行加密才进行传递。设备在注册过程中，可由设备的拥有者通过线下安全传输待注册设备的设备信息，确保设备信息的安全性。

1) 用户账号安全性。对于安全目录创建者而言，设备管理模块验证其身份合法性的方式是为其产生用户 ticket，并加密存储。首先，对于外部入侵者而言，即使获取了加密后的 ticket，也由于没有系统秘密密钥而无法解密出有效的 ticket 信息。另外，由于存储的内容不直接是用户的账号密码信息，因此，即使外部入侵者得到 ticket 信息，由于使用的哈希函数的不可逆性，也无法反向获得用户的账号和密码信息，最大程度地保护了用户账号信息的安全性。

2) 文件数据的安全性。无论在源设备端还是共享设备，一个 SFS 结构的文件首先需要验证其文件头的文件消息摘要码。当且仅当文件的完整性和正确性被保证时，设备才对文件进行解密，有效地

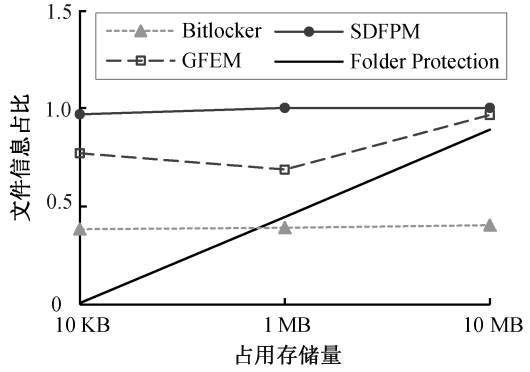

图 5　不同文件保护工具的文件信息占比
Fig. 5　Information rate comparison of different tools

防止攻击者进行文件伪造。任何一个 SFS 结构的文件存储的都是文件内容的密文，即使攻击者获得一个被保护的文件，也只能提取出文件加密密钥的密文和内容的密文。获取文件加密密钥的唯一条件是授权设备，且其本身的信息正确可靠，才能还原出文件解密密钥，进而解密受保护的文件。因此，在设备硬件信息的提取方法保密的前提下，攻击者即使获得 SFS，也无法读取其中的有效信息。此外，秘密系统密钥的使用也增强了数据的安全性。

4 结论

信息安全，尤其是记录用户个人隐私的文件安全和小型企业存储在不可信介质中的文件资料安全[5]，越来越受到人们的重视。目前主流的文件保护软件无法对同一用户多个设备上的文件进行保护和安全共享。本文提出一种基于安全目录的文件保护机制 SDFPM，将存储在安全目录中的隐私文件自动加密保护并封装，使得所有格式的文件都能用其进行保护。同时，SDFPM 能够实现受保护文件在多个设备上的安全共享。实验和分析表明，SDFPM 具有良好的安全性、响应速率和存储性能。

在未来的工作中，我们将进一步实现对安全目录中的子目录进行保护的功能，以支持在安全目录中创建、修改、删除子目录，对子目录内存储的文件进行实时加密保护与管理。此外，受限于个人存储设备硬件容量的限制，越来越多的人选择将信息存储在云平台和移动办公软件上，期望 SDFPM 将来能够与云存储服务进行无缝衔接，实现对不完全可信的云平台上的文件进行自动保护。

参考文献

[1] 俞银燕，汤帜. 数字版权保护技术研究综述. 计算机学报, 2005, 28(12): 1957–1968

[2] Cross D B, Leach P J. File system operation and digital rights management (DRM): US, Patent No. 8, 640, 256 [P]. 2014–01–28

[3] Kumar C A. Designing role-based access control using formal concept analysis. Security and Communication Networks, 2013, 6(3): 373–383

[4] Park J, Sandhu R. Towards usage control models: beyond traditional access control // ACM Symposium on Access Control MODELS & Technologies. Monterey, 2002: 57–64

[5] Park J, Sandhu R. The UCON ABC usage control model. ACM Transactions on Information and System Security (TISSEC), 2004, 7(1): 128–174

[6] 刘庆云，沙泓州，李世明，等. 一种基于量化用户和服务的大规模网络访问控制方法. 计算机学报, 2014, 37(5): 1195–1205

[7] Xiong Z, Xu J Y, Wang G J, et al. UCON application model based on role and rule-engine. Computer Engineering & Design, 2013, 34(3): 831–836

[8] Huang G X. Analysis of kernel technology about Windows encrypt file system. Computer & Information Technology, 2005, 13(4): 1–5

[9] Hamada M. Secure anonymously authenticated and traceable enterprise DRM system. International Journal of Computer Applications, 2015, 126(3): 1–9

[10] Zhao X, Borders K, Prakash A. Towards protecting sensitive files in a compromised system // Third IEEE International Security in Storage Workshop. San Francisco, 2005: 21–28

[11] Park J, Sandhu R. The UCON ABC usage control model. ACM Transactions on Information and System Security (TISSEC), 2004, 7(1): 128–174