计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2020,29(11):151−156 [doi: 10.15888/j.cnki.csa.007673]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

DPAPI 与 RSA 混合加密算法①

金美玉, 汤亚玲, 张学锋

(安徽工业大学 计算机科学与技术学院, 马鞍山 243002)
通讯作者: 金美玉, E-mail: 2979595130@qq.com

摘　要: 为了应对软件被恶意复制及非法利用等相关问题, 对具有知识产权的软件进行安全授权是保障软件安全的有效手段. 在软件授权过程中, 对授权数据采用高度安全的加密算法尤为重要. 文中提出一种新的授权加密方法, 即 DPAPI 加密算法和 RSA 数字签名算法的混合加密算法. 该算法利用 DPAPI 加密算法加密客户端申请授权的信息, 在实现加密的同时保证软件授权的正确性, 再利用 RSA 数字签名算法对服务器端的授权信息进行数字签名, 以保证授权信息的不可伪造性. 通过对该混合加密算法的验证可知, 该算法在软件授权过程中具有一定的可行性.

关键词: DPAPI 加密算法; RSA 数字签名算法; 软件授权; 安全性; 可行性

引用格式: 金美玉,汤亚玲,张学锋.DPAPI 与 RSA 混合加密算法.计算机系统应用,2020,29(11):151–156. http://www.c-s-a.org.cn/1003-3254/7673.html

DPAPI and RSA Hybrid Encryption Algorithm

JIN Mei-Yu, TANG Ya-Ling, ZHANG Xue-Feng

(School of Computer Science and Technology, Anhui University of Technology, Maanshan 243002, China)

Abstract: In order to cope with problems related to malicious copying and illegal use of software, security authorization of software with intellectual property rights is an effective means to ensure software security. In the software authorization process, it is particularly important to use a highly secure encryption algorithm for the authorization data. This study proposes a new authorization encryption method, that is, a hybrid encryption algorithm of DPAPI encryption algorithm and RSA digital signature algorithm. This algorithm uses the DPAPI encryption algorithm to encrypt the client's application for authorization information, while ensuring the encryption and the correctness of the software authorization at the same time, and then uses the RSA digital signature algorithm to digitally sign the server-side authorization information to ensure the unforgeability of the authorization information. The verification of the hybrid encryption algorithm shows that the algorithm has certain feasibility in the software authorization process.

Key words: DPAPI encryption algorithm; RSA digital signature algorithm; software licensing; security; feasibility

软件授权是指软件开发者利用相关加密算法及计算机技术来维护软件知识产权, 增加软件被恶意复制盗用的难度、延长软件被暴力破解的时间的一种技术手段[1]. 利用软件授权来控制用户对软件的使用是目前维护软件知识产权最普遍的方式. 随着互联网技术的发展, 计算机软件与人们日常生活的联系越来越紧密, 伴随而来的软件知识产权保护问题也越来越重要[2]. 目前互联网上已经出现很多未授权用户可以非法使用软件甚至恶意复制软件的行为, 说明软件授权的安全性还有待提高. 为了防止软件被恶意复制及非法盗用, 针

① 基金项目: 安徽省教育厅自然科学基金重大项目 (KJ2017ZD05)
Foundation item: Major Program of Natural Science Foundation of Education Bureau, Anhui Province (KJ2017ZD05)
收稿时间: 2020-03-05; 修改时间: 2020-04-10, 2020-04-24; 采用时间: 2020-05-13; csa 在线出版时间: 2020-10-29

对软件授权的安全问题, 提出一种新的软件授权加密方法即 DPAPI 加密算法与 RSA 数字签名算法混合的加密算法. 为以后软件授权技术的研发提供一个新的授权思路, 进一步提高软件授权过程中的安全性.

1 DPAPI 加密算法

DPAPI (Data Protection API) 是 Microsoft 随 Windows 操作系统一起提供的对数据进行加解密的一种特殊保护接口, 又称为数据保护应用程序编程接口[3], 具有快速加密解密, 不用管理密钥, 防止离线暴力破解等优点. DPAPI 加密算法提供两种加密机制, 分别是加密内存流机制和加密硬盘数据机制[4]. 本文采用第二种加密机制. DPAPI 加密算法的硬盘数据加密机制由加密函数 CryptProtectData() 和解密函数 CryptUnProtectData() 进行实现[5], 且其与 Windows 系统用户紧密关联. 若用系统用户调用加密函数, 则加密后的数据只能由调用加密函数的同一系统用户进行解密, 且系统用户不能调用解密函数解密其他系统用户的 DPAPI 加密数据[6], 这也是 DPAPI 加密算法能防止离线暴力破解的原因. 本文也是利用 DPAPI 加密算法的这一特性作为软件是否正确授权的参考. DPAPI 加密算法的流程如图 1 所示.

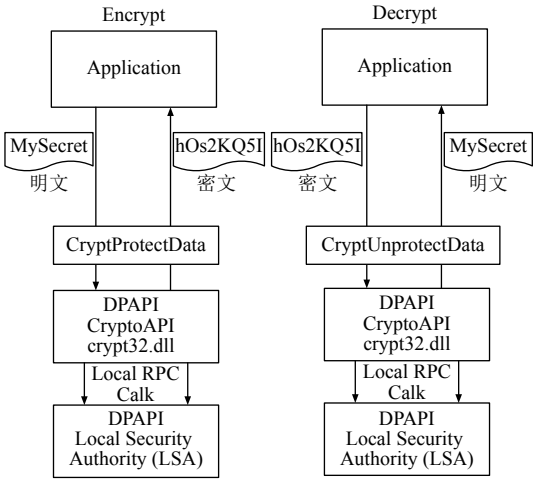

图 1 DPAPI 加解密流程图

2 RSA 数字签名算法

RSA 加密算法是目前应用广泛、影响较大的公钥密码算法[7], 它能抵抗迄今为止已知的大多数密码攻击,已被 ISO 推荐为公钥数据加密标准[8]. 其安全性是基于数论中大整数 (两个素数的乘积) 因式分解的困难程度, 即素数取值越大, 对大整数做因式分解的难度就越高, 密钥破解难度相对也就越大, 算法也就更安全. 换一句话说, 分解大数因式愈困难, RSA 算法也就愈安全. 除此之外, RSA 算法的通信双方不需要提前传递密钥, 安全性能高, 且其数学原理简单, 在工程应用中比较容易实现, 既可以用来对数据进行加密又可以用来对信息发送方进行身份的认证, 常与其他加密算法结合使用.

RSA 数字签名算法是经由 RSA 公钥密码算法发展而来的[9], RSA 数字签名算法使用私钥对数据进行签名 (也可称之为加密), 使用公钥对数据进行验证 (也可称之为解密), 如图 2 所示. 进行 RSA 数字签名有以下 3 种目的, 一是为了确定接收方接收的数据确实是由发送方对数据进行签名并发送的, 即发送方身份认证. 二是经过数字签名的数据可保证自身的数据完整性, 即不可篡改伪造. 三是数据经过数字签名之后, 发送方不能拒绝承认数据及签名是己方发送的, 即不能抵赖不承认. 因为数字签名的特性就是表征数据特点的, 如果数据被篡改或伪造, 同样也会致使签名的更改, 不同的数据将得到不同的签名结果 (因签名是私钥及哈希加密数据后的结果), 所以数字签名在保证数据的安全、完整及真实性等方面有着非常重要的作用. 对数据进行 RSA 数字签名的过程[10] 如图 2.

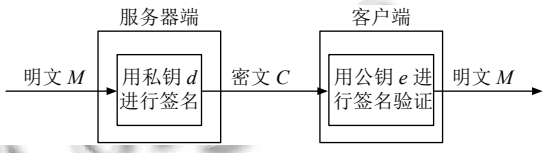

图 2 RSA 数字签名过程

(1) 生成密钥: (发送方即服务器端) 取大素数 p 和 q, 两者互质且保密; 计算大整数$n = p*q$及欧拉函数 $\varphi(n) = (p-1)*(q-1)$; 取整数 e,使得 e 满足条件$1 < e < \varphi(n)$, 且$(\varphi(n), e) = 1$; 再取 e 的逆元为 d, 使得$d = e^{-1} \pmod{\varphi(n)}$; 取 (e,n) 为公钥, (d,n) 为私钥.

(2) 签名: 服务器端发送签名消息 M 到客户端, 先计算$S = Sig(M) = M^d \pmod{n}$再将 S 作为服务器端的数字签名附加到消息 M 上.

(3) 验证: 若客户端 (即接收方) 需要验证服务器端对消息 M 的数字签名 S,则计算: $M' = S^e \pmod{n}$, 比较 M'与 M 是否相等, 若$M' = M$, 则表明签名 S 确实是服务器端对消息 M 的签名, 否则签名 S 不是由服务器端生成的, 是伪造的.

3 DPAPI 与 RSA 混合加密算法

3.1 DPAPI 与 RSA 混合加密算法的提出

从 Windows 2000 到更高版本的操作系统中都提供 DPAPI 加密算法, 用户或操作系统程序都可以直接调用 DPAPI 接口函数来对数据进行加密解密. 由于 DPAPI 加密算法具有简单易用、加密强度大、不用管理密钥 (操作系统管理密钥) 的特性使得大量应用程序都采用 DPAPI 加密算法来加密用户的重要数据. 如远程桌面及 IE 浏览器的自动登录密码、邮箱 (Outlook) 账号密码等[6]. 其次在进行 DPAPI 加密的过程中, 若加密之前与计算机关联, 并使用用户口令或计算机凭据加密数据, 则解密操作只能在实施加密操作的计算机上执行, 即使传输的数据被拦截, 第三方也很难实现离线的暴力破解[4]. 最后, DPAPI 加密算法的内部结构及操作原理一直处于封闭状态并未在微软官方公布, 这也在一定程度上保证了 DPAPI 加密算法的安全性. 因此, 使用 DPAPI 加密算法加密数据相当于使用了 Windows、应用程序两个级别的数据保护[4].

RSA 数字签名算法的安全性高、理论性简单、易于实现, 是目前应用最为广泛的数字签名算法. 因其公钥和私钥的不同, 无法由其中一个密钥推断出另一个密钥[11], 这是 RSA 算法的安全性所在. 目前攻击 RSA 数字签名算法的方法主要有算法攻击和存在性伪造攻击[9]. 其中算法攻击的本质是指破解 RSA 的私钥, 也就是对大整数进行因式分解, 而大整数因式分解的安全性到目前为止还没有得到理论上的完全证明, 许多专家一致认为 RSA 算法的安全性等同于分解大整数的难度, 并且与 RSA 相关的变体算法也已经被证明相当于大数分解[12], 所以为了防止该类攻击, 通常使用较长的模数长度; 而存在性攻击是指攻击者或通信双方之外的第三方仿冒数据的有效签名, 以此来谋取公钥和私钥, 从而达到破解数字签名的目的. 为了防止数字签名遭受以上攻击, RSA 数字签名算法通常采用 RSA 填充技术进行抵抗, 该技术又称为概率签名标准技术. RSA 填充技术是一种基于 RSA 密码体制的签名方案, 它可以很好的防范上述攻击, 在 RSA 数字签名算法中应用广泛[9].

综上所述, DPAPI 加密算法和 RSA 数字签名算法的安全性及保密性都有一定的保证. 若两种加密算法能够很好地结合, 就可以设计出一个 DPAPI 与 RSA 混合的加密算法, 既能保证数据接收方的正确性, 又可以保证数据发送方的信息不会被仿冒. 将该混合算法应用于软件的授权过程, 更好地保证授权安全, 是本文研究的目的所在. DPAPI 与 RSA 混合加密算法的架构图如图 3.

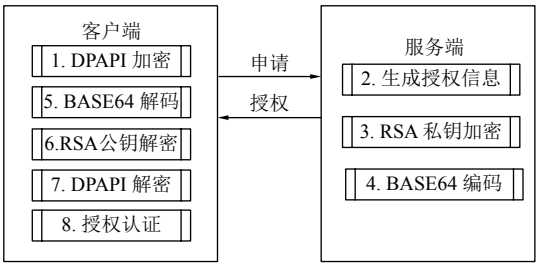

图 3 DPAPI 与 RSA 混合算法协同工作示意图

3.2 DPAPI 与 RSA 混合加密算法的原理

DPAPI 加密算法一机一用的特点对于软件授权来说是判断是否正确授权的重要参考, 其次, 利用 RSA 数字签名算法的私钥对授权信息进行加密可以保证通信双方之外的第三方无法伪造该信息, 也是证明发送方信息真实性的依据[13]. 本文将 DPAPI 加密算法与 RSA 数字签名算法进行混合并应用于软件授权上. DPAPI 与 RSA 混合应用示意图如图 4 所示.

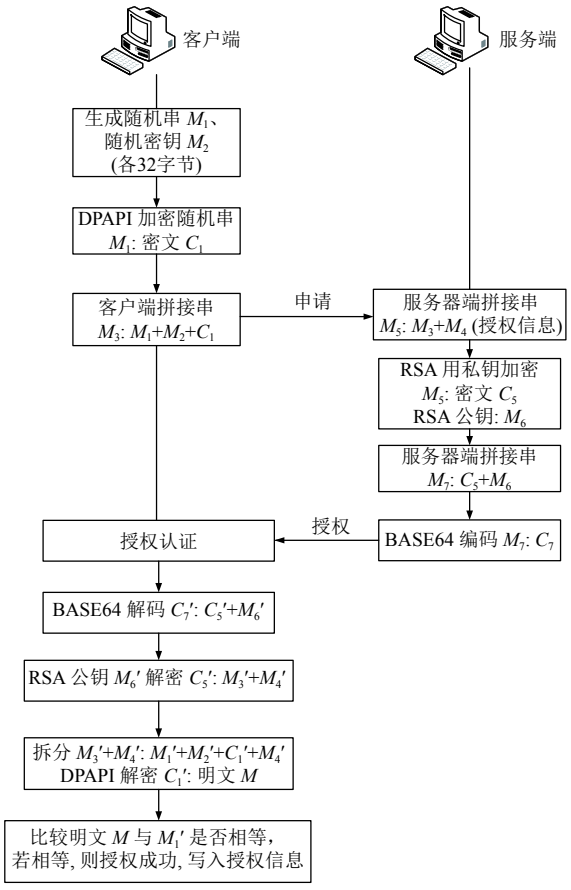

图 4 混合加密算法示意图

具体实现原理如下:

(1) 客户端通过自定义函数随机生成随机串M_1和随机密钥M_2, 长度均为 256 位.

(2) 客户端通过 CryptProtectData() 函数对随机串M_1进行 DPAPI 加密得到密文C_1, 密文长度为 402 位.

(3) 拼接随机串M_1、随机密钥M_2、密文C_1, 得到客户端拼接串M_3, 长度为 256+256+402=914 位.

(4) 客户端将拼接串M_3发送给服务器, 服务器端在客户端拼接串M_3中加入授权信息M_4, 此处的授权信息为软件使用截止日期, 如: 2088-08-08, 得到服务器端拼接串$M_5 : M_3 + M_4$, 长度为 914+10=924 位.

(5) 服务器端将拼接串M_5通过 RSA 私钥进行加密, 实验中 RSA 相关函数的使用来自 OpenSSL 软件库包. 其中服务器端使用 RSA_private_encrypt() 函数来对拼接串M_5进行私钥加密, 从而获得密文C_5.

(6) 服务器端将密文C_5与 RSA 公钥M_6拼接成字符串$M_7 : C_5 + M_6$, 并通过 BASE64 编码得到字符串C_7.

(7) 服务器端发送授权文件 (字符串C_7) 到客户端, 客户端接收并进行授权认证, 执行 BASE64 解码, 得到$C_7' : C_5' + M_6'$, 再由 RSA 公钥M_6'通过 RSA_public_decrypt() 函数对C_5'进行解密得到服务器端拼接串$M_5' : M_3' + M_4'$.

(8) 客户端拆分$M_3' + M_4'$得到$M_1' + M_2' + C_1' + M_4'$, 由随机密钥$M_2'$通过 CryptUnprotectData() 函数对密文C_1'进行解密得到明文随机串 M. 比较随机串 M 和 M_1', 若相等, 写入授权信息M_4', 否则授权失败.

4 DPAPI 与 RSA 混合加密算法的实现

利用实验室工程应用中的软件对本文设计的混合加密算法进行验证 (使用 3 台机器进行验证). 机器 A 和 B 分别作为客户端 1 和 2 来安装软件并利用客户端 1 进行软件授权的申请, 机器 C 作为服务器端用于返回授权文件. 如果在客户端 1 上进行了加密, 那么在客户端 1 上能够解密且在客户端 2 上解密失败, 则视为验证通过. 3 台机器之间通过传送授权文件进行通信, 进而验证混合加密算法的正确性及可行性, 其中授权文件中包含用于比较验证的密文数据及授权信息.

由于 DPAPI 加密算法在 Windows 2000 及以上的操作系统中都有函数功能接口, 用户及操作系统程序都可以直接使用它. 例如在 VC 环境下, 引用头文件 <Windows.h>和<wincrypt.h>, 就可以调用 DPAPI 的加密解密函数. 对于 RSA 相关函数, 该实验通过引用 OpenSSL 软件库包来提供. 搭建过程如下: 在 Windows 环境下安装和编译 OpenSSL, 编译成功后再配置 VC 的 OpenSSL 开发环境, 即把 OpenSSL 库中 include 文件下的 OpenSSL 文件拷贝至 VC 安装目录下的 include 文件夹中, 再将 lib 文件夹下的库文件拷贝至 VC 的 lib 文件夹中, 最后, 在 VC 的项目工程里设置工程属性将库添加进去, 测试成功就可以使用 RSA 的相关函数了. 如图 5 所示, 客户端生成随机串和随机密钥.

图 5 随机串与随机密钥

根据图 4 可知, 混合加密算法主要有 DPAPI 加密、RSA 私钥加密、BASE64 编码、BASE64 解码、RSA 公钥解密、DPAPI 解密操作 6 个步骤. 因此在实现 DPAPI 与 RSA 混合加密算法时, 要用到相应加密算法的函数, 具体有以下 4 个核心函数 (由于参数较多此处省略介绍, 详情可查看官方文档):

1) DPAPI_IMP BOOL CryptProtectData();

2) int RSA_private_encrypt();

3) int RSA_public_decrypt();

4) DPAPI_IMP BOOL CryptUnprotectData().

函数 1 的作用是将参数 pDataIn 的实参值转换为 DPAPI 加密的密文, 即客户端提交的明文 (随机串M_1) 转换成 DPAPI 加密后的密文C_1. 同时设置参数 dwFlags 的取值为 CRYPTPROTECT_LOCAL_MACHINE, 将加密的数据与当前计算机关联,而不是与单个用户关联. 关联后, 该加密数据只能在该机器上进行解密操作, 否则解密出的数据与原先的明文数据不同, 进而不能获得授权.

函数 2 的功能是对服务器端的拼接串M_5 (图 6) 进行 RSA 私钥加密, 即进行数字签名. 服务器端先接收客户端发送的信息M_3并通过字符串拼接函数将授权信息M_4与M_3进行拼接, 然后再进行 RSA 私钥加密. 本文通过调用 PEM_write_bio_RSAPrivateKey() 函数来生成私钥 (图 7), 再利用 PEM_read_bio_RSAPrivateKey() 函数将 RSA 私钥信息存储在参数 rsa 中. 在进行 RSA 私钥加密之前需将字符串进行分割, 因 padding 参数采

用 RSA_PKCS1_PADDING 填充方式,采用该模式进行 RSA 加密对密文的长度存在限制,密文最大长度为 117 位.此处采用分段加密,将服务器端拼接串 M_5 分成 $\lceil 924 \div 117 \rceil = 8$ 段,其中 $\lceil x \rceil$ 表示不小于 x 的最小整数. 最后, 对 RSA 私钥加密后的信息再进行二次处理,通过 Base64 编码得到字符串 C_7, 增加了字符串的安全性.

图 6 服务器端拼接串

图 7 PEM_write_bio_RSAPrivateKey() 函数生成私钥

客户端对字符串 C_5' 进行 RSA 公钥解密操作即函数 3 的功能. 客户端接收授权文件 (C_7) 并进行授权认证, 利用 Base64 执行解码操作, 再利用 RSA 公钥对解码后的信息 C_5' 进行解密操作 (图 8). 本文通过调用 PEM_write_bio_RSAPublicKey() 函数来生成公钥 (如图 9), 再利用 PEM_read_bio_RSAPublicKey() 函数将 RSA 公钥存储在函数 3 的参数 rsa 中, 且参数 padding 的取值与 RSA 私钥加密函数的 padding 参数相同, 也采用 RSA_PKCS1_PADDING 填充方式. 本文中对 RSA 公钥解密后的字符串再进行拆分操作,便于进行 DPAPI 解密操作.

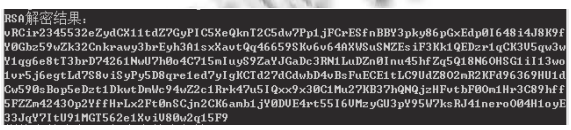

图 8 RSA 公钥解密结果

图 9 PEM_write_bio_RSAPublicKey() 函数生成公钥

客户端对字符串 C_1' 进行 DPAPI 解密操作即函数 4 的功能. 函数 3 执行过后, 客户端再利用字符串拆分函数对 RSA 公钥解密后的信息进行字符串拆分保存, 将拆分后的信息通过函数 4 进行解密, 解密后的信息再与拆分保存的字符串通过验证函数进行对比验证 (如图 10), 验证函数返回值为 true 时才可进行软件授权信息的写入.

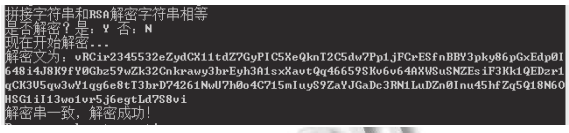

图 10 验证结果

基于上述对混合加密算法的验证可知, 相较于单独使用 DPAPI 或者 RSA 加密算法进行软件授权, 该混合加密算法的安全性是增大的, 且混合加密算法的两层加解密过程是分别调用官方库函数进行的, 不会过多的消耗计算资源, 还进一步增大了第三方破解该加密方案的复杂度. 其次, 因 DPAPI 和 RSA 加密算法本身具有较强的抗攻击特性, 理论上对于常见的攻击方式, 混合加密算法可做到很好的抵御. 最后, 本文提出的混合加密方案是在服务端加密和客户端加解密过程来完成一次软件授权的, 通过实验可知, 两层加密解密过程是有效的, 即该混合加密算法是可行的.

5 结束语

首先, 文章介绍了 DPAPI 加密算法和 RSA 数字签名算法的原理及安全性依据. 其次, 介绍了 DPAPI 与 RSA 混合加密算法在软件授权过程中的应用原理. 最后, 在 VC 环境下实现并验证了这一混合算法. 该算法结合了二者的优点. DPAPI 加密算法用于对客户端申请授权的信息进行加密, 在一机一用的前提下保证授权的正确性; 再用 RSA 算法的私钥加密服务器端的授权信息保证了授权信息不会被仿冒. 由此可见, DPAPI 加密算法与 RSA 数字签名算法的混合, 能更好的保证软件在授权过程中的正确性和安全性, 同时, 该混合算法在实验室的工程应用中得到了验证, 表明该算法具有一定的可行性, 可以作为一种软件授权方案进行推广和使用.

参考文献

1 王琴琴, 郭师虹. 软件授权技术的研究. 计算机技术与发展, 2012, 22(9): 235–238.

2 欧阳雪, 周寰, 邓锦洲, 等. 一种面向软件生命周期的授权保护系统设计与实现. 计算机工程与科学, 2013, 35(4): 59–64. [doi: 10.3969/j.issn.1007-130X.2013.04.011]
3 DPAPI 加密算法. http://passcape.com/index.php?section=blog&cmddetails&id20.
4 谢利东, 李志坚. 智能客户端本地数据加密方案. 信息通信, 2014, (5): 90–91.
5 Burzstein E, Picod JM. Recovering windows secrets and EFS certificates offline. Proceedings of the 4th USENIX Conference on Offensive Technologies. Berkeley, CA, USA. 2010. 1–8.
6 钱镜洁, 林艺滨, 陈江勇. DPAPI 离线解密方法及其取证应用. 信息网络安全, 2013, (7): 39–43. [doi: 10.3969/j.issn.1671-1122.2013.07.010]
7 Rivest RL, Shamir A, Adleman L. A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM, 1978, 21(2): 120–126. [doi: 10.1145/359340.359342]
8 翁云翔. 基于 DES 和 RSA 的混合加密算法研究与设计. 电子设计工程, 2016, 24(17): 42–44, 47. [doi: 10.3969/j.issn.1674-6236.2016.17.013]
9 谭德林, 李均利. 一种具有数字签名的二维码技术. 计算机技术与发展, 2018, 28(3): 143–145. [doi: 10.3969/j.issn.1673-629X.2018.03.030]
10 肖振久, 胡驰, 陈虹. 改进的 RSA 算法在数字签名中的应用. 计算机工程与应用, 2014, 50(17): 106–109. [doi: 10.3778/j.issn.1002-8331.1210-0044]
11 王煜, 朱明, 夏演. 非对称加密算法在身份认证中的应用研究. 计算机技术与发展, 2020, 30(1): 94–98. [doi: 10.3969/j.issn.1673-629X.2020.01.017]
12 侯佩, 寇雅楠, 黄利斌. 混合加密体制在数字签名中的应用. 计算机工程与设计, 2011, 32(6): 1942–1945.
13 肖振久, 胡驰, 陈虹. 四素数 RSA 数字签名算法的研究与实现. 计算机应用, 2013, 33(5): 1374–1377.