数字签名标准(DSS)的理论研究与实现

唐益慰,孙知信
(南京邮电大学 计算机学院,江苏 南京 210003)

摘 要:随着信息和信息技术的发展,电子数据交换逐步成为人们交换的主要形式,密码在信息安全中的应用将会不断拓宽,信息安全对密码的依赖会越来越大。介绍了密码体系中数字签名DSS,主要阐述了它的快速实现方法和主要步骤,包括DSS的主要算法:大素数的寻找,蒙特卡罗算法,高次幂的模指数剩余计算,模逆的计算。给出了DSS在计算机中的模拟过程,并对其安全性进行了分析和说明。

关键词:数字签名标准;验证;公钥;私钥;密码
中图分类号:TP309.7 **文献标识码**:A **文章编号**:1673-629X(2006)09-0233-03

Theory of Digital Signature Standard Technology and Its Applications

TANG Yi-wei, SUN Zhi-xin
(Coll. of Computer Sci. and Techn., Nanjing Univ. of Posts and Telecommunications, Nanjing 210003, China)

Abstract: The electronics data exchange become the main form that people exchange gradually. The cryptography will open widely and continuously in application of the information safety. The information safety would be more and more counting on the cryptography. This article describes one of the cipher systems: data signature system (DSS). Primarily discuss the quick method to realize DSS and its primary steps which includes the primary arithmetic of the DSS: finding big prime number, Monte Carlo arithmetic, the residual calculate to mod exponent with high power, the arithmetic to power adverse. This article gives the computer programs to simulate the process DSS. And it analyses and explains the security of DSS.

Key words: digital signature standard; signature; public key; private key; cipher

1 数字签名标准的原理

1.1 数字签名标准概述

数字签名标准(Digital Signature Standard, DSS)[1,2]是美国国家标准与技术研究所(NIST)于1991年8月公布的,1994年5月19日正式公布,1994年12月正式作为美国联邦信息处理标准FIPS 186颁布(该标准后来经过了一些修改,目前的标准称为FIPS 186-2)。DSS中所采用的算法通常称为DSA(Digital Signature Algorithm)。

DSS从诞生以来,一直争议不断,这其中的主要原因除了安全性之外,另外一个重要的原因是NIST没有采用已经工业界得到广泛应用并成为了事实上标准的RSA数字签名体制,而另行开发了DSA[3,4]算法,这样,许多公司投入的用于获取、实现RSA算法的大量资金将可能白白流走(特别是持有RSA算法专利权的美国RSA数据安全公司),工业界当然不愿意看到这样的局面;同时,NIST后来承认:DSA算法是由NSA(美国国家安全局)指导设计的,这更加重了公众关于NSA可能在算法中设置陷门,以便在必要的时候可以由NSA对DSA算法进行破译的猜疑,但这些争议实际上大多政治意义强于技术意义,从目前对DSA算法的攻击来看,没有充分的证据表明DSA算法在安全性上存在很大的安全弱点。DSA已经在许多数字签名标准中得到推荐使用,除了联邦信息处理标准FIPS 186-2外,IEEE的P1363标准中数字签名标准也推荐使用DSA等算法。

1.2 DSS的签名与验证过程

1.2.1 数字签名步骤

DSS中规定使用了安全的散列算法(SHA)[5,6],图1所示是DSS的签名与验证过程,说明如下:

(1)发送方将发送信息原文用SHA函数编码,产生固定长度的数字摘要。

(2)发送方用自己的私钥对摘要加密,形成数字签名,附在原文后面。

(3)发送方产生通信密钥,用它对带有数字签名信息进行加密,传到接受方。

(4)发送方用接受方的公钥对自己的通信密钥进行加密后,传到接受方。接受方收到加密后的通信密钥,用自己的私钥对其进行解密,得到发送方的通信密钥。

收稿日期:2005-11-29
作者简介:唐益慰(1982-),男,浙江人,硕士研究生,研究方向为计算机网络与安全;孙知信,教授,研究方向为计算机网络与安全、软件工程。

(5)接受方用发送方的通信密钥对收到的经加密签名原文解密,得到数字签名和原文。

(6)接受方用发送方公钥对数字签名解密,得到摘要;同时将原文 SHA 函数编码,产生另一个摘要。

(7)接受方将两个摘要比较,若一样,说明信息没有被破坏或篡改。

DSS 基于离散对数问题,它可以看作是 ELGAMAL 数字签名体系的一个变种。以下将描述整个算法。

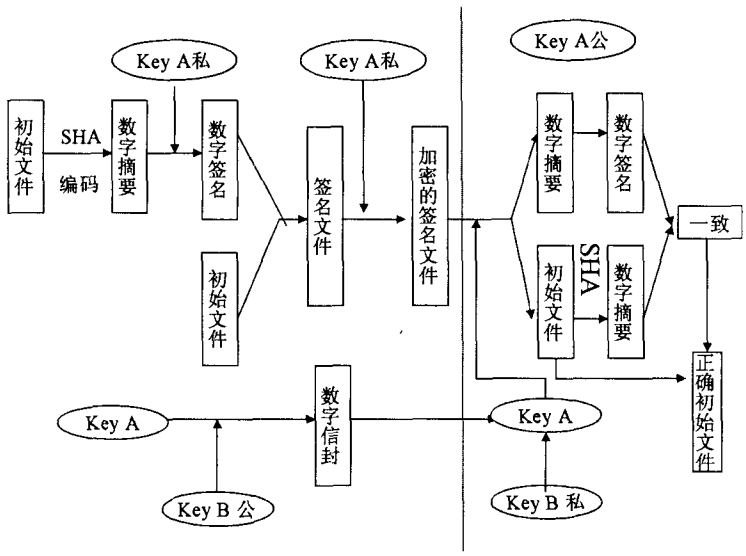

图 1 数字签名体系的详细步骤

1.2.2 数字签名系统参数

DSS 中使用的参数包括系统的公共参数与每一个用户的密钥:

系统公共参数[7,8]:

(1)大素数 P,$2^{L-1} < P < 2^L$,其中 $521 \leqslant L \leqslant 1024$,且 L 被 64 整除(即 P 为 512 ~ 1024 二进制位长并且长度为 64 所整除的大素数);

(2)$P-1$ 的素因子 Q,Q 的长度为 160 比特;

(3)$\alpha = h^{(p-1)/q} \pmod p$,其中,$1 < H < (P-1)$,并使 $h^{(p-1)/q} \bmod p > 1$。

每一个用户(假设用户为 U)的公钥 c_u 与密钥 m_u:

用户 U 的公开密钥与秘密密钥按照如下方式产生:随机选择一个整数 m_u,$0 < m_u < q$,并计算 $c = \alpha^{m_u} \pmod p$,用户 U 公开密钥并保存整数 m_u 作为自己的秘密密钥。

1.2.3 签名过程

假设用户 A 对消息 M 进行数字签名后发送给接受者 B。

首先,用户 A 选择一个随机数 R,$0 < R < Q$,并计算:

$$R = (\alpha^r \pmod p) \pmod q$$

$$S = r^{-1}(\mathrm{SHA}(M) + M_A * R) \pmod q \qquad (1)$$

其中,式中的 r^{-1} 是随机数 R 的模 Q 乘法逆,$r * r^{-1} \equiv 1 \pmod q$;SHA($M$) 代表使用安全散列算法 M 进行散列[9]。

然后,用户 A 将(R,S)作为自己对待消息 M 签名,即 Sig(M) = (R,S),并随消息 M 一起发送给接受者 B。

1.2.4 验证过程

用户 B 在接受到用户 A 发送来的消息 M 及签名(R,S)后,计算:

$$W = S^{-1} \pmod q$$

$$U_1 = (\mathrm{SHA}(M) \cdot W) \pmod q$$

$$U_2 = (R \cdot W) \pmod q$$

$$V = ((\alpha^{U_1} * c_{A^2}^{U_2}) \pmod p) \pmod q \qquad (2)$$

并判断是否有 $V = R$。由于

$$V = ((\alpha^{U_1} * \alpha^{U_2}) \pmod p) \pmod q$$

$$V \equiv \alpha^r \pmod q$$

$$V \equiv R \qquad (3)$$

因此,如果 $V = R$,则说明签名有效;否则签名有可能是伪造的。

下面就给出一个 DSA 算法的签名与验证过程。取素数 $Q = 23$,$P = 47$,并取 $\alpha = 17^2 \bmod 47 = 7$。假设用户 A 选择了整数 $m_A = 10$ 作为自己的秘密签名密钥,因此,用户 A 的公开签名密钥 $C_A = \alpha^{m_A} \pmod p = 7^{10} \pmod{47} = 32$,用户 A 对 SHA($M$) = 15 进行运算得到签名。首先,用户 A 产生随机数 $R = 19$,根据(扩展的)欧几里得算法,$r^{-1} = 17 \pmod{23}$。

然后,用户 A 计算 $R = (\alpha^r \pmod p) \pmod q = 12$,接着,用户 A 将计算 S,

$$S = r^{-1} * (\mathrm{SHA}(M) + m_A * R) \pmod q$$

$$S = 17 \cdot (15 + 10 \cdot 12) \pmod{23}$$

$$S = 18$$

用户 A 把元组 (R,S) = (12,18) 作为自己对 SHA(M) = 15 的消息 M 的签名。

验证签名(R,S) = (12,18) 时,验证方需要计算

$$W = S^{-1} \pmod q = 9$$

$$U_1 = (\mathrm{SHA}(M) * W) \pmod q = 20$$

$$U_2 = (R \cdot W) \pmod q = 16$$

$$V = ((\alpha^{U_1} * c_{A^2}^{U_2}) \pmod p) \pmod q = 12$$

此时,$V = R = 12$,说明签名是有效的;否则 $V \neq R$,签名无效。基于 DSS 的数字签名流程如图 2 所示。

在具体实现中,DSS 系统需要完成下面几个算法:大素数的产生算法(包括素数的产生和验证),模 n 求逆的算法,模 n 的大数幂乘的快速算法,安全散列算法(SHA),和数字签名 DSS 算法。由于密码技术的发展,产生了各种各样对素数进行操作的算法,解决了对大素数的产生×运算和存储的复杂性.在研究不同算法的基础上,文中将选取其中优秀的算法,来实现 DDS 数字签名。

2 DSS 系统测试

文中分别实现了 DSS 算法的签名和验证过程。系统可以分别对任何文件或一段消息进行数字签名和验证,并

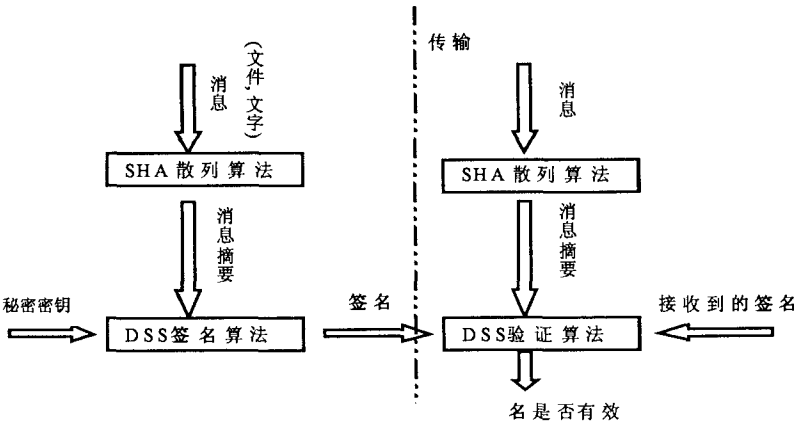

图2　基于DSS的数字签名流程图

且实现了签名功能与验证功能的分离,用户可以通过系统得到自己的惟一密钥和公钥,通过密钥的签名和公钥的解密,来确定用户与文件的关系,具有不可抵赖性。系统运行过程如下所示:

启动DDS系统,如果用户还没有自己的密钥,那么系统可以运用大素数算法产生一系列随机的160bit密钥提供用户选择,用户选择一个惟一密钥来作为自己的惟一标记。例如用户A选择了一个自己的私钥如下:

private key is:

x = 424F, 1C86, 92E2, 7123, 2889, 0C99, D60D, CC9C,6415,451Bh

当用户A选定了一个密钥后,系统将自动产生一个对应的公钥:

the public key is:

y = 692B, BF47, 98FC, C6F0, 209A, B82A, CE64, 8E70,B4DA,9ECE,79DA,007F,4013,956A,A818,B7D2, DE5D,2D9F,8EDE,CF84,8837,2134,1226,30C2,C244, 77F5,421A,3CDE,D589,07A9,7B07,1A28h

并且产生一系列大素数供系统签名所需,在实际应用中,用户可以不用了解具体的大素数产生情况。

the big numbers system created:

p is a 512bit prime number ,p = A680,405C,6CA9, 7627,1A63,8C56,536A,B506,9DE7,7067,69BF,87BF, D3BB,D53F,C14C,90DB,0F98,2755,DB48,58FA,D21F, 7D67,C158,DB96,FB10,E4D5,AC32,6B5D,4588,91A4, 307E,F25Bh

q is a 160bit prime number ,q = D53D,291F,BC37, 94CB,A23C,1D5A,3464,5CF2,8484,A2FDh

g = h^(p-1)/q mod p. 1<h<p

g = ADC, 84F6, A644, E36C, D110, E91D, D67C, 4A6D, 6DDA, ABC2, 5DBA, 9867, 9B59, 22F1, B1B1, 8DCB,E387,3735,5B55,07CB,AC3B,2695,C98B,6F66, 865B,BA2C,830D,141E,53EF,4430,520D,81A0h

k is a random number :0<k<q ,k = 5E25,C722, 7AF4,41EC,9987,01A0,C9E1,1293,D744,87C2h

上面的p,q,g,k之间的关系在上文都已做了详细说明。

大数系统产生之后,就可以进行DSS签名了。首先可以对一段文本进行签名。

input the words you want to sign: tangyiwei

using SHA method to hash it: hash = D4B3, DCDD, 1E56, 114A, 52C7, 719D, 05D8,DF97,7FF4,AA17h

Press "s" to sign the words:s =[K^(-1)(H(m)+xr)]mod q

s = 26D, B15D, 9A54, 138D, EB5B, 5F2E,19D5,2E97,36A9,7D34h

r=(g^k mod p)mod q) =B243,D4AF,B78E,881C, 18CE,A808,2F36,A1C4,D282,3071h

系统首先对文本运用SHA散列算法进行散列,然后运用私钥进行签名。得到结果 s,r。相应的算法见上文。

当终端用户需要认证的时候,通过公钥 y 进行。具体如下所示:

为了验证数字签名的不可抵赖性,首先输入与私钥 x 不对应的公钥 y:

y = E28, B5E0, 3850, EB54, E68E, 2AC6, A7FE, 8A5D,87FE,94B7,8912,9438,5BC7,FA35,ED75,80C8, CEC3, AECA, C2EA, 4E18, D251, DD31, 27D9, 14CD, 2A00,4D65,1A2F,965C,9251,772D,7BE8,B1C6h

v= (g^u1 * y^u2 mod p) mod q= 7B1E,C29A, A223,A39B,32B5,BD72,C9CB,7C0A,38E0,04A0h

v≠r FAILED valid signature verification

当输入与私钥对应的公钥时,产生的结果如下:

y = 692B, BF47, 98FC, C6F0, 209A, B82A, CE64, 8E70,B4DA,9ECE,79DA,007F,4013,956A,A818,B7D2, DE5D,2D9F,8EDE,CF84,8837,2134,1226,30C2,C244, 77F5,421A,3CDE,D589,07A9,7B07,1A28h

v=(g^u1 * y^u2 mod p) mod q= B243,D4AF,B78E, 881C,18CE,A808,2F36,A1C4,D282,3071h

v=r PASSED valid signature verification

系统认证成功。对于对文件的签名与对消息的加密运行结果相似,这里就不详述了。S从上面的测试中可以得到,运用DSS算法,只要用户保护好自己的密钥,就可以使用户的信息不可被其他非法用户修改,并且一旦用户运用自己的私钥进行了签名,其对于这个信息也具有不可抵赖性。完全满足与公钥签名的规范要求。

程序实现了DSS算法所要求的内容:大数的形成,大数四则混合运算,大数求模运算,大素数的概率检测算法,SHA散列算法,欧几里得算法。在每次的运行中,能随机地产生DSS所需要的各个参数,并能对用户所输入的ASCII码和数据文件进行数字签名。稍微地改动程序,可

(下转第239页)

理时,环境变量 permitprint = "false",因此,在计算机屏幕上是看不到电子公章的。但当打印文件时,permitprint = "true",这样,具有读者存取权限的用户只有通过打印机才能够打印出带有电子公章的标准红头文件。

同时,对院办秘书这样的用户还设计了与其他用户不同的功能,其中包括公文打印预览、发送公文、进入起草状态、查阅公文签收状态、公文打印等。通过这些安全性控制,进一步加强了电子公章的安全性保护,从而也保证了文件传输安全、可靠。

2.4.3　打印控制及打印状态监视系统

本系统对公文的合法打印以及打印份数进行了设计,并实现了一些安全控制机制。设计各单位打印份数的初始值为"2",即变量 typenum = 2。如果打印份数大于2份,则有再申请打印份数的提示。在设计中,设置环境变量"permitprint"初始值为"false ",打印签名变量 printname 为"",当用户打印文件并签名后,环境变量"permitprint"为"true ",打印签名变量 printname 不为空,打印份数加1,即 Printnumber + 1。这时方可打印带有电子印章的公文正文。但对一般用户,由于电子公章域有禁止预览读的控制,"读者"用户在计算机屏幕上仍始终看不到电子印章。

另外还设计了一个打印状态监视库 print。打印状态监视库的管理员设为"院办秘书",其为打开 print 库、查询各个文件的打印结果、修改打印份数的权限。在该数据库中,同时还设计了打印控制视图和打印查询视图。打印控制视图主要是发文时间,打印查询视图主要是创建时间。两个视图均可以对各单位的秘书收文状况进行动态监视。

3　结　论

OA系统的网络层和数据库应用层的安全性建设决定整个系统的安全等级,而高安全性和高可用性一直是文中研究的重点。设计了基于C/S和B/S混合模式的OA系统,提出了综合利用 Lotus Domino/Notes 本身具有的安全机制、网络的安全技术和操作系统本身的安全特性,建立OA系统安全控制机制的模型。本模型经实验证明具有高安全性、高可用性。但高安全性必然是效率的损失。如何在OA系统里满足高安全性、高可用性要求的同时,提高文件处理的效率,将是下一步将要讨论的问题。

参考文献:

[1] 邴志刚,郑　君,储　健,等. 基于 Domino/Notes 的高校OAS解决方案[J]. 天津职业技术师范学院学报,2003(1):19-22.

[2] 张秋余,袁占亭,冯　涛. 办公自动化系统的设计与安全策略研究[J]. 兰州理工大学学报,2004(2):82-85.

[3] 余冬梅,刘密霞,冯　涛. 网络安全系统设计的研究[J]. 微机发展,2004,14(2):125-127.

[4] 李文印,吴　迪,叶润国. OA系统安全性设计及实现[J]. 吉林大学学报(信息科学版), 2003(4): 21-26.

[5] 郭雪梅,陈家晖,张江洋. OA系统安全防范的几点方法[J]. 广东自动化与信息工程,2004(4):43-45.

[6] 马康玉,刘　超,陈剑波. 生物识别技术在法院办公自动化中的应用-电子印章系统开发[J]. 微机发展,2003,13(5):52-54.

(上接第235页)

以使 DSS 签名的复杂度增加,主要是大数的长度增加。最大可以产生1024位的有效签名。不过解密加密时系统所付出的代价将会比较大。

3　总　结

介绍了密码体系中数字签名标准 DSS,主要阐述了它的快速实现方法和主要步骤,包括 DSS 的主要算法:大素数测试米勒-勒宾算法,高次幂的模指数剩余计算,模逆的计算。给出了 DSS 在计算机中的模拟过程,并对其安全性进行了分析和说明。通过 DSS 加解密系统程序的测试和运行,证明了 DSS 算法在防窜改、防抵赖等的功能,从中得到其在电子商务、保密传输方面的广泛应用。随着当今世界信息化的发展,数字签名技术作为网络信息安全的重要方面,必将得到进一步的发展。

参考文献:

[1] Harn L, Mehta M, Wen-Jung Hsin. Integrating Diffie-Hellman key exchange into the digital signature algorithm (DSA) [J]. Communications Letters, IEEE, 2004, 8(3): 198-200.

[2] Nel J J, Kuhn G J. Generation of keys for use with the digital signature standard (DSS), Communications and Signal Processing[A]. Proceedings of the 1993 IEEE South African Symposium[C]. Jan Smuts Airport: Institute of Electrical and Electronics Engineers, Rand Afrikaans University, 1993.6-11.

[3] Arazi B. Integrating a key distribution procedure into the digital signature standard[J]. Electronics Letters, 1993, 29(11): 966-967.

[4] 陈少真,李大兴. 基于变形 DSA 的有效群签名[J]. 计算机工程与设计,2004,25(3):323-326.

[5] Schneier B. 应用密码学——协议、算法和C源程序[Z]. 吴世忠等译. 成都:国防保密重点实验室,1996.

[6] 陈鲁生,沈世镒. 现代密码学[M]. 北京:科学出版社,2002.

[7] 卡茨安 H. 标准数据加密算法[M]. 陈太一,屠世桢译. 北京:人民邮电出版社,1983.

[8] 杨义先,林须端. 编码密码学[M]. 北京:人民邮电出版社,1992.

[9] 卢铁成. 信息加密技术[M]. 成都:四川科学技术出版社,1989.