基于 hash 函数的一次群签名模型*

高晓东[1,2]，杨亚涛[2]，李子臣[2]
(1. 西安电子科技大学 通信工程学院，西安 710071；2. 北京电子科技学院，北京 100070)

摘 要：针对基于大整数的素数分解和离散对数问题的传统数字签名不能抵抗量子时代量子计算的攻击问题，提出一种基于 hash 函数的一次群签名模型。该模型基于 hash 函数的单向性，由 hash 运算完成密钥生成、签名和验证，获得了更高的效率，并且可有效抵抗量子时代量子计算的攻击。通过实验，对签名模型进行验证，效率比 ECC(密钥长度为 224)高 21 倍，可达 RSA-2048 的 102 倍。

关键词：量子攻击；一次群签名；单向函数；hash 运算
中图分类号：TP309 **文献标志码**：A **文章编号**：1001-3695(2012)07-2665-03
doi:10.3969/j.issn.1001-3695.2012.07.072

One-time group signature model based on hash function

GAO Xiao-dong[1,2], YANG Ya-tao[2], LI Zi-chen[2]
(1. *School of Telecommunication Engineering, Xidian University, Xi'an* 710071, *China*; 2. *Beijing Electronic Science & Technology Institute, Beijing* 100070, *China*)

Abstract: Since traditional digital signature based on the big integer prime decomposition and discrete logarithm problem can't resist attack in quantum computing, this paper proposed a signature model based on the hash function. The security of this model is based on the hash function is unidirectional. Its key generation, signature and verification are completed by hash computing. This model could help achieve higher efficiency, and resist attack in quantum computing effectively. Proved by experiments, its efficiency is 21 times higher than ECC (a key length for 224), and can reach 102 times higher than RSA-2048.
Key words: quantum attacked; one-time group signature; one-way function; hash computing

0 引言

群数字签名是由 Chaum 等人[1]于 1991 年提出的。群签名方案要求群中的任何一个成员都可以对某一个消息进行群签名。签名的验证只需要用群公钥(或群的某一公开身份)来验证即可，同时要求群签名具有不可伪造性，任何人都不能代表群中某一成员对消息进行签名。群签名能够隐藏组织的内部结构，广泛运用在政治、军事和商业领域中。一次签名是一种特殊的数字签名，其基本思想是利用无陷门单向函数对消息进行签名。这种签名方案的安全性是基于无陷门的单向函数，它生成和验证签名的速度很快。一次群签名是在群签名的过程中暴露部分私钥。每签名一次更换一次密钥，来保证签名的绝对安全性。一次性签名和群体签名结合起来可以充分利用两者的优势。

hash 函数是一种无陷门的单向函数，它把任意长度的输入信息，通过散列算法，变换成固定长度的消息摘要。人们若对消息摘要进行数字签名不但可以提高签名的效率，而且还可以保证信息完整性和防止隐私信息泄露。可见，hash 函数在数字签名中起着非常重要的作用。但是它在通常的数字签名中主要是求消息摘要，而真正的签名算法是基于 RSA、ElGamal、ECC 等公钥密码体制[2,3]。目前，这些数字签名体制比较成熟，已经广泛应用于各种电子政务、电子商务活动中。但这些传统的数字签名体制无法抵抗量子时代量子计算对它的攻击。

Lamport[4]于 1979 年提出了一种基于单向函数的一次数字签名方案。拉开了利用单向函数的单向性进行数字签名的序幕，但在这个方案中签名和验证的密钥量比较大，最后得到的签名的数据量也很大，阻碍着该方案的实际运用。Chen 等人[5]运用 Seberry 提出的方法构造了一个 hash 轮函数，并且利用此 hash 轮函数和 Girault 提出的一种自我认证的公钥密码算法构造并实现了一种新的被称为 H-S DSA 的数字签名算法，在此用 hash 轮函数只是为了让消息在签名之前进行 hash 变换，以防止替换攻击和同态攻击，而真正的签名算法还是传统的公钥密码体制。Zhou 等人[6]为了提高数字签名的安全性提出了一种基于混沌的 hash 函数，但是签名算法仍运用 RSA 公钥密码体制，签名的安全性还是基于大数分解的困难性，无法抵抗后量子时代的量子攻击。

针对此问题，提出了一种基于 hash 函数的一次群签名模型。该模型在整个签名过程中只运用 hash 函数，通过 hash 运算来完成密钥生成、签名过程和验证过程，而没有运用 RSA、ElGamal、ECC 等公钥密码体制，其安全性不是基于大整数的素数分解和离散对数问题，而是基于 hash 函数的单向性，由于安全 hash 函数是通过循环迭代一种安全的特殊结构来增加扩散和混淆，能有效地抵抗后量子时代量子计算的攻击。

收稿日期：2011-12-01；**修回日期**：2012-01-15 **基金项目**：国家自然科学基金资助项目(61070219)；北京电子科技学院信息安全重点实验室资助项目

作者简介：高晓东(1984-)，男，甘肃定西人，硕士研究生，主要研究方向为信息安全(gaoxiaodong_191@163.com)；杨亚涛(1978-)，男，讲师，博士，主要研究方向为网络与信息安全；李子臣(1965-)，男，教授，博士，主要研究方向为网络与信息安全、密码学.

1 基于 hash 函数的一次群签名

基于安全的 hash 函数,本文提出了一种一次群签名模型。该模型不但在求消息摘要的时候运用 hash 函数,而且在密钥生成算法、签名算法和验证算法中也仅运用 hash 运算,没有用到 RSA、ElGamal 等公钥密码算法。在签名过程中群成员借助可信中心机构来生成签名密钥和验证公钥,通过 hash 运算实现签名的整个过程。

1.1 符号说明

表 1 本文使用符号

符号	符号说明
m	群成员个数
q	成员密钥中 n bit 字符串的个数
$d\in\{0,1\}^n$	文件 M 的 n bit 消息摘要值
$\lceil n/m\rceil$	小于等于 n/m 的最大整数
$\{0,1\}^n$	长为 n bit 字符串
$\{0,1\}^{(q,n)}$	q 个 n bit 长的字符串
$X_k\in\{0,1\}^{(q,n)}$	第 k 个成员的成员密钥
$Y\in\{0,1\}^n$	群公钥
$P\in\{0,1\}^{(n,n)}$	中间变量
n	消息摘要长度
H	$\{0,1\}^n\rightarrow\{0,1\}^n$ 的 hash 函数
$H(x)$	对 x 进行一次 hash 运算
$\oplus$	按位模二加法运算
$\{0,1\}^*$	任意长的 0,1 字符串
$\{0,1\}^{(n,n)}$	n 个 n bit 长的 0,1 字符串
$X\in\{0,1\}^{(n,n)}$	群私钥(签名密钥)
$S\in\{0,1\}^{(n,n)}$	对文件 M 的签名
$O\in\{0\}^n$	n bit 的全 0 字符串

1.2 群密钥对的生成

1.2.1 签名密钥的产生

群中每个成员任意选取 $q=\lfloor n/m\rceil$ 个长为 n bit 的 0、1 字符串作为成员密钥 $X_k\in\{0,1\}^{(q,n)}(1\leqslant k\leqslant m)$。然后把这 m 个成员密钥通过安全信道传递给可信中心机构(TA),这 m 个成员密钥一共有 $m\times q$ 个 n bit 长的二元字符串。这样可信中心机构就得到 n 个长为 n bit 的二元字符串(若 m 不能整除 n,则 TA 再任意选取 $n-m\times\lfloor n/m\rfloor$ 个长为 n bit 的字符串)分别为 $x_0,x_1,\cdots,x_{n-1}$。TA 将这 n 个长为 n bit 的字符串组成群私钥,作为签名密钥 $X=(x_0,x_1,\cdots,x_{n-1})\in\{0,1\}^{(n,n)}$。其中 $x_i(0\leqslant i\leqslant n-1)$ 均为群成员传给 TA 的长为 n bit 的二元字符串(若 m 不能整除 n,则 n 个字符串 $x_i(0\leqslant i\leqslant n-1)$ 中还包括 $n-m\times\lfloor n/m\rfloor$ 个 TA 任意选取的长为 n bit 的字符串)。

1.2.2 群公钥的产生

群公钥是对签名密钥进行 n 次的 hash 运算得到的。TA 根据签名密钥 $X=(x_0,x_1,\cdots,x_{n-1})$ 从 $i=0$ 到 $n-1$ 计算 $y_i=H(x_i)$。最后得出群公钥 $Y=H(y_0\oplus y_1\oplus\cdots\oplus y_{n-1})$。群公钥公开,作为验证签名算法中的验证密钥。

1.3 签名过程

群中的每一个成员都可以代表群对文件进行签名,若由第 k 个群成员来签名,则签名过程如下。

对一个文件 $M\in\{0,1\}^*$ 进行签名,先求出文件 M 的消息摘要 $d=(d_0,d_1,\cdots,d_{n-1})\in\{0,1\}^n$。

第 k 个成员把自己的成员密钥 X_k 通过安全信道传递给 TA,如果 X_k 是群私钥 $X=(x_0,x_1,\cdots,x_{n-1})$ 中的 q 个 n bit 长的字符串,则认为它为合法的群成员,否则为非法群成员。一旦 TA 认为它是合法的群成员,就将除了 X_k 以外的部分密钥传递给该成员。TA 传输给群成员的部分密钥是 X_k 中除去 X_k 以外的 $n-q$ 个长为 n bit 的字符串。TA 通过安全信道把部分密钥传输给第 k 个群成员后,该群成员根据收到的部分密钥和自己的成员密钥恢复签名密钥 $X=(x_0,x_1,\cdots,x_{n-1})$(签名密钥中 x_i 的顺序对签名没有影响,虽然不同的顺序所得到的签名不同,但是在验证过程中的结果是一致的)。最后该成员运用签名密钥 X 对文件 M 的消息摘要 $d=(d_0,d_1,\cdots,d_{n-1})$ 进行数字签名。签名为 $S=(s_0,s_1,\cdots,s_{n-1})\in\{0,1\}^{(n,n)}$。

其中,

$$s_i=\begin{cases}x_i & d_i=o\\ H(x_i) & d_i=1\end{cases}\quad(0\leqslant i\leqslant n-1)$$

这个签名 $S=(s_0,s_1,\cdots,s_{n-1})$ 是文件 M 的消息摘要 $d=(d_0,d_1,\cdots,d_{n-1})$ 的函数。如果在 d 中的第 i 位 $d_i=0$ 时,则在这个签名中的第 i 个字符串为 x_i;否则为 $H(x_i)$。对文件 M 的签名 $S=(s_0,s_1,\cdots,s_{n-1})$ 是 n 个长为 n bit 的二元字符串。

1.4 验证过程

验证者首先计算文件 M 的消息摘要 $d=(d_0,d_1,\cdots,d_{n-1})$。然后用群公钥 Y 对上述数字签名 $S=(s_0,s_1,\cdots,s_{n-1})$ 进行运算得 $P=(p_0\oplus p_1\oplus\cdots\oplus p_{n-1})$。

其中,

$$p_i=\begin{cases}H(s_i) & d_i=0\\ s_i & d_i=1\end{cases}\quad(0\leqslant i\leqslant n-1)$$

计算 $H(P)\oplus Y$ 验证结果是否等于 $O\in\{0\}^n$,若等于 O 则认为该次对文件 M 的签名有效;否则无效,说明该签名不是对文件 M 的合法签名。具体的签名和验证过程如图 1 所示。

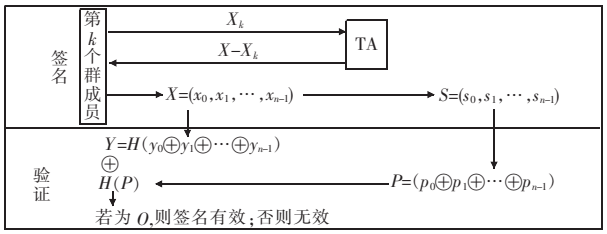

图 1 签名和验证过程

2 正确性证明

该签名方案在签名和验证过程中先求出文件 M 的消息摘要 $d=(d_0,d_1,\cdots,d_{n-1})$。群中的任何一个成员可以代替群对文件 M 的消息摘要 d 进行签名。假设第 k 个成员代替群对文件 M 签名。该成员恢复签名密钥 $X=(x_0,x_1,\cdots,x_{n-1})$。并对文件 M 的消息摘要 d 进行数字签名。签名为 $S=(s_0,s_1,\cdots,s_{n-1})$。

其中,

$$s_i=\begin{cases}x_i & d_i=0\\ H(x_i) & d_i=1\end{cases}\quad(0\leqslant i\leqslant n-1)$$

验证者用群公钥 Y 对上述数字签名 $S=(s_0,s_1,\cdots,s_{n-1})$ 进行验证,经过运算得到 $P=(p_0\oplus p_1\oplus\cdots\oplus p_{n-1})$。

其中,

$$p_i=\begin{cases}H(s_i) & d_i=0\\ s_i & d_i=1\end{cases}\quad(0\leqslant i\leqslant n-1)$$

当 $d_i=0$ 时,$p_i=H(s_i)=H(x_i)$;

当 $d_i=1$ 时,$p_i=s_i=H(x_i)$;

所以,

$$H(P)=H(p_0\oplus p_1\oplus\cdots\oplus p_{n-1})=$$

$$H(H(x_0)\oplus H(x_1)\oplus\cdots\oplus H(x_{n-1}))$$

又因为群公钥:

$$Y=H(y_0\oplus y_1\oplus\cdots\oplus y_{n-1})=$$
$$H(H(x_0)\oplus H(x_1)\oplus\cdots\oplus H(x_{n-1}))$$

所以可以得出 $H(P)\oplus Y=O$。因此只要签名合法必然满足 $H(P)\oplus Y=O$。

3　安全性分析

1)密钥安全性　本方案的安全性完全取决于选取的 hash 函数的安全性。密钥是 TA 通过成员密钥计算得到的。TA 计算出群签名密钥之后通过安全信道传输给签名者。在 TA 给群成员传输签名密钥时只传输该成员不知道的部分密钥。这样更加保证了签名密钥在传输过程中的安全性。

群公钥 $Y=H(y_0\oplus y_1\oplus\cdots\oplus y_{n-1})$ 是签名私钥 $X=(x_0,x_1,\cdots,x_{n-1})$ 通过 hash 运算得到的,其中 $y_i=H(x_i)(0\leqslant i\leqslant n-1)$。若选取安全的 hash 函数(如 SHA-3 候选算法 BLAKE),由于安全的 hash 函数具有不可逆性,因此,任何人从群公钥出发计算出签名密钥是困难的。

2)不可替代性　其他任何人包括其他群成员不知道签名密钥 $X=(x_0,x_1,\cdots,x_{n-1})$,而签名算法为: $S=(s_0,s_1,\cdots,s_{n-1})$ 其中,

$$s_i=\begin{cases}x_i & d_i=0\\ H(x_i) & d_i=1\end{cases}\quad(0\leqslant i\leqslant n-1)$$

可见,签名是利用签名密钥和信息的消息摘要通过 hash 运算得到的,其他任何人不可能知道签名密钥也就无法伪造群签名。

4　效率测量

实验选用 BLAKE 算法作为 hash 函数。目前最安全、最有竞争力的 hash 函数是 NIST(美国国家标准技术研究所)在全世界征集 hash 函数在第二轮候选算法筛选中获胜的五个候选算法(BLAKE、Grostl、JH、Keccak、Skein)。BLAKE 是这五个获胜候选算法之一,以实现速度快、抗差分等特性而著名[7]。设消息摘要的长度为 n bit,在密钥生成中需要对签名密钥进行 $n+1$次 hash 运算和 $n-1$ 次按位模二加法运算得到群公钥。本次测试中密钥生成时间还包括通过产生随机数得到签名密钥的时间。然后运用此密钥对对一个固定摘要进行签名和验证过程,测试实现这些过程所需要的时间。本次测试是在 Pentium 43.06 GHz 处理器、1 GB 内存、Linux Ubuntu 10.04 LTS 操作系统下测试的。BLAKE 有四种类型,分别是 BLAKE224、BLAKE256、BLAKE384、BLAKE512,对应的消息摘要为 224、256、384、512[8]。测试中对于 n 取 224、256、384、512 时,分别由 BLAKE224、BLAKE256、BLAKE384、BLAKE512 来实现密钥生成、签名和验证过程,并且通过十次测试求平均值得到最终结果。通过测试,基于 hash 函数的数字签名中各阶段所需要的时间如表 2 所示 。

表 2　签名各阶段所需要的时间

hash 函数	n	密钥产生/ms	签名/ms	验证/ms	总时间/ms
BLAKE224	224	0.796	0.266	0.350	1.412
BLAKE256	256	0.940	0.499	0.572	2.011
BLAKE384	384	4.416	2.328	2.521	9.265
BLAKE512	512	6.246	3.091	3.486	12.823

目前相对安全的 RSA-2048、ECC(密钥长度为 224)的加/解密算法至少需要几十毫秒的时间。密钥对生成所需要的时间更长,至少要两三百毫秒。整个过程所花费的时间 RSA-2048 为 1 312 ms,ECC(密钥长度为 224)为 281 ms[9]。RSA 和 ECC 是公钥密码体制,如果用公钥加密,而用私钥解密则就成了签名算法,利用 RSA 和 ECC 的签名算法是使用它们的加/解密算法的逆过程,则用 RSA 和 ECC 的签名和验证所用的时间与对应的解密和加密的时间相同。总的签名时间和总加密时间是相同的。文献[9]中测试出的数据是在 PC 机上(CPU: PentiumDual E2180 2 GHz,内存:1 GB)测试的跟前面测试基于 hash 函数的数字签名所使用微机的配置差不多,测试的数据具有可比性。可以推出在整个利用 RSA 和 ECC 的数字签名过程中总共使用的时间如下:RSA-2048 为 1 312 ms,ECC(密钥长度为 224)为 281 ms。基于 hash 函数的数字签名所需总时间最高不到 13 ms。相比之下,基于 hash 函数的数字签名实现效率比 ECC 高 21 倍,可达 RSA 的 102 倍,并且可以抵抗未来量子分解算法对它的攻击。

5　结束语

传统的数字签名体制是利用 RSA、ElGamal、ECC 等公钥密码体制,它们的安全性是基于大整数的素分解和离散对数问题,无法抵抗量子计算的攻击。针对此问题,本文提出一种基于 hash 函数的一次群签名模型,该模型中的密钥生成、签名和验证都通过 hash 运算实现。通过实验,对签名模型进行验证,效率比 ECC(密钥长度为 224)高 21 倍,可达 RSA-2048 的 102 倍。该模型获得了更高的效率,并且可以抵抗量子时代量子计算的攻击。因此,这种更安全有效的基于 hash 函数的数字签名体制将有很大的发展前景。

参考文献:

[1] CHAUM D, Van HEYST E. Group signatures[C]//Proc of Workshop on the Theory and Application of Gryptographic Techniques. New York: Springer-Verlag, 1991: 257-265.

[2] FANG De-jian, WANG Na, LIU Cheng-lian. An enhanced RSA-based partially blind signature[C]//Proc of International Conference on Computer and Communication Technologies in Agriculture Engineering. 2010: 565-567.

[3] WANG Xue-ming, DONG Yu-rong. Threshold group signature scheme with privilege subjects based on ECC[C]//Proc of International Conference on Communications and Intelligence Information Security. 2010: 84-87.

[4] LAMPORT L. Constructing digital signatures from a one way function SRI-CSL-98[R]. SRI International Computer Science Laboratory, 1979.

[5] CHEN Hai-peng, SHEN Xuan-jing, WEI Wei. Digital signature algorithm based on hash round function and self-certified public key system[C]//Proc of the 1st International Workshop on ETCS. 2009: 618-624.

[6] ZHOU Chuan-hua, ZHU Ge-mei, ZHAO Bao-hua, *et al*. Study of one-way hash function to digital signature technology[C]//Proc of International Conference on Computational Intelligence and Security. 2006: 1503-1506.

[7] MING Mao, QIANG He, ZHEN Shao-kun. Security analysis of BLAKE-32 based on differential properties[C]//Proc of International Conference on ICCIS. 2010: 783-786.

[8] AUMASSON J P, HENZEN L, MEIER W, *et al*. SHA-3 Proposal BLAKE [EB/OL]. (2011-06-20)[2011-12-01]. http://www.131002.net/blake/.

[9] YUAN Yang-tao, LIU Quan, LI Fen. A design of certificate authority based on elliptic curve cryptography[C]//Proc of the 9th International Symposium on Distributed Computing and Applications to Business, Engineering and Science. 2010: 454-457.