ICS 35.040
L 80

中华人民共和国国家标准

GB/T 17902.2—2005/ISO/IEC 14888-2:1999

信息技术　安全技术　带附录的数字签名　第2部分:基于身份的机制

Information technology—Security techniques—Digital signatures with appendix—Part 2:Identity-based mechanisms

(ISO/IEC 14888-2:1999,IDT)

2005-04-19 发布　　2005-10-01 实施

中华人民共和国国家质量监督检验检疫总局
中国国家标准化管理委员会　发布

中 华 人 民 共 和 国
国 家 标 准
信息技术 安全技术 带附录的数字签名
第 2 部分:基于身份的机制
GB/T 17902.2—2005/ISO/IEC 14888-2:1999
*
中 国 标 准 出 版 社 出 版 发 行
北京西城区复兴门外三里河北街 16 号
邮政编码:100045
http://www.spc.net.cn
电话:63787337、63787447
2005 年 8 月第一版 2005 年 8 月电子版制作
*
书号:155066·1-23069

目　　次

前　言

GB/T 17902《信息技术　安全技术　带附录的数字签名》由以下几个部分组成：

第1部分：概述；

第2部分：基于身份的机制；

第3部分：基于证书的机制。

本部分为GB/T 17902的第2部分，等同采用国际标准ISO/IEC 14888-2:1999《信息技术　安全技术　带附录的数字签名　第2部分：基于身份的机制》(英文版)。

本部分的附录A和附录B是资料性附录。

本部分由中华人民共和国信息产业部提出；

本部分由全国信息安全标准化技术委员会归口；

本部分由中国电子技术标准化研究所、信息安全国家重点实验室起草。

本部分主要起草人：叶茅枫、陈星、罗锋盈、胡磊、叶顶锋、张振峰、黄家英。

信息技术　安全技术　带附录的数字签名
第2部分:基于身份的机制

1　范围

GB/T 17902规定了任意长度消息的带附录的各种数字签名机制。它适用于提供实体鉴别、数据始发鉴别、抗抵赖和数据完整性的方案。

GB/T 17902的本部分规定了任意长度消息的带附录的基于身份的数字签名机制的签名和验证过程的总的结构和基本过程。

2　规范性引用文件

下列文件中的条款通过GB/T 17902的本部分的引用而成为本部分的条款。凡是注日期的引用文件,其随后所有的修改单(不包括勘误的内容)或修订版均不适用于本部分,然而,鼓励根据本部分达成协议的各方研究是否可使用这些文件的最新版本。凡是不注日期的引用文件,其最新版本适用于本部分。

GB 15851—1995　信息技术　安全技术　带消息恢复的数字签名方案(idt ISO/IEC 9796:1991)

GB/T 17902.1—1999　信息技术　安全技术　带附录的数字签名　第1部分:概述

3　概述

数字签名的验证需要签名实体的验证密钥。所以,验证方必须把正确的验证密钥与签名实体,或更准确地讲,与签名实体的(部分)标识数据关联起来。如果这种联系是验证密钥自身所固有的,这种方案被称作“基于身份的”。

本部分中定义的基于身份的方案的密钥生成过程包括一个可信第三方。这个可信第三方有个秘密参数——密钥生成指数,它用于导出其他实体的签名密钥。签名密钥的秘密性无条件地依赖于密钥生成指数的秘密性。

在基于身份的签名的验证中,需要两个参数。第一个参数为域验证指数,它对所有实体来说是共同的;而第二个参数为签名实体的验证密钥,它对每个实体而言是特定的。本部分定义的基于身份的机制中,实体的验证密钥是使用一个公共函数直接从实体的标识数据中得到的。

带附录的基于身份的签名机制是个随机化机制的例子,如GB/T 17902.1—1999所描述。数字签名和验证过程描述遵循GB/T 17902.1—1999第10章定义的一般过程。特别地,本标准使用了GB/T 17902.1—1999提供的一般需求条件、定义和符号。

在下列过程的详细说明中定义了带附录的基于身份的数字签名机制。它们是:

a)　密钥生成过程;

b)　签名过程;

c)　验证过程。

4　术语和定义

下列术语和定义适用于GB/T 17902的本部分:

4.1

域模数　domain modulus

域参数，它是一个正整数，是只有可信第三方才知道的两个不同的素数相乘产生的。

4.2

域验证指数　domain verification exponent

一个域参数，它是一个正整数。

4.3

密钥生成指数　key generation exponent

一个只有可信第三方才知道的正整数。

4.4

公钥导出函数　public key derivation function

一个域参数，它的功能是将比特串映射成正整数。

注 1：这个函数用于将实体标识数据转换成实体验证密钥，并符合下列两个性质：

a) 要找出任何一对映射成同一输出的两个不同的输入数据，在计算上是不可行的。

b) 随机选取数值 Y，Y 在函数值域范围内的概率小到可以忽略；或者对给定的输出数据，找出可映射成该输出的输入，在计算上是不可行的。

注 2：可忽略性与计算上的不可行性依赖于具体的安全要求和环境。

4.5

可信第三方　trusted third party

一个安全机构或者它的代理，在安全相关活动方面，被其他实体信任。

5　符号

GB/T 17902.1—1999 中确立的以及下列符号适用于 GB/T 17902 的本部分：

D	密钥生成指数
I	标识数据
N	域模数
P，Q	素数
V	域验证指数
y	公钥导出函数
lcm(A，B)	使得 $C \bmod A = 0$ 和 $C \bmod B = 0$ 的最小正整数 C

6　密钥生成过程

基于身份的签名机制的密钥生成过程是 GB 15851—1995 附录 A 中规定的签名方案的一个应用。它由下列两个过程组成：

a) 生成域参数；

b) 生成签名密钥。

应将一个实体选定为可信第三方。使用它自己的秘密，可信第三方为每个实体生成该实体的私有签名密钥。该密钥是实体的标识数据的函数。

在某些情况下，也许需要进行域参数和密钥的确认。但这超出了 GB/T 17902 的本部分的范围。

6.1　生成域参数

建立域时，这个过程只执行一次。

可信第三方生成域验证指数 V 和域模数 N。域验证指数应选择为一个奇整数。域模数应为两个大素数 P 和 Q 的乘积，$P-1$ 和 $Q-1$ 与 V 互素。更进一步，可信第三方确定密钥生成指数 D，它使得 $DV-1$ 为 lcm ($P-1$，$Q-1$)的倍数的最小正整数。

明确地讲，D 应使得：

$U^{DV} \bmod N = U$ 对所有整数 U，$0<U<N$

N 和 V 为公开域参数。可信第三方保存 D 为自己使用。其他实体通过 N 和 V 将 D 计算出来是不可行的。

注：N 和 V 的值应选择大到足以满足特定的域安全的需要。N 的长度通常在 1 024 比特和 2 048 比特之间变化，而 V 的长度推荐至少为 80 比特。

设 T，T' 和 L 是使得 $T - T' = LV$ 的整数，X 是一个签名密钥，而 Y 是相应的如 6.2 中定义的验证密钥。则 $X^{T'} \equiv X^T \cdot Y^L \bmod N$。

给定一个对应于 T 的签名(R, S)，则可以通过计算 $S' = S \cdot Y^L \bmod N$（见 7.4）生成一个对应于 T' 的签名(R, S')。因此，必须选择足够大的 V 以使给定 T 后，一个随机选取的 T' 满足于 $T - T' \equiv 0 \bmod V$ 的概率足够小。

基于身份的带附录的签名机制的域参数的设定还包括公钥导出函数 y，它用于将一个签名实体的标识数据转换成一个小于 N 的正整数。

6.2 生成验证密钥和签名密钥

每个用户实体有其唯一的标识数据。要为一个标识数据为 I 的实体生成私有签名密钥，可信第三方首先从公钥导出函数 y 和标识数据 I 计算出验证密钥 Y：

$$Y = y(I)$$

注：只要 P 和 Q 足够大，则 Y 等于零或者等于 P 或 Q 的整数倍的概率可以忽略。

可信授权方计算私有签名密钥 X 为：

$$X \equiv Y^{-D} \bmod N$$

作为密钥生成过程的一个结果，标识数据 I 的实体拥有一个签名密钥 X，它满足方程：

$$X^V \cdot y(I) \bmod N \equiv 1$$

在基于身份的签名机制中，通过计算 $Y = y(I)$，验证方从签名实体的标识数据中得到了签名实体的验证密钥的信息。当验证进行后，可以将其存储备以后使用。

7 签名过程

在本章中描述基于身份的签名机制的签名过程。该机制是随机化的并遵从 GB/T 17902.1—1999 描述的随机化签名机制的一般模型。签名过程由下列步骤组成：

a) 生成预签名；

b) 准备消息；

c) 计算证据；

d) 计算签名。

在此过程中，签名者使用它的签名密钥 X 和公共域参数 N 和 V。

此过程的输出为签名 Σ，它由两部分 R 和 S 组成。签名实体可选择地生成一个含有实体的标识数据文本字段。签名和可选的文本字段形成附录，它由签名者附加到消息后。

7.1 生成预签名

基于身份的签名机制的预签名过程由下面两个步骤组成：

a) 生成随机数 K；

b) 计算预签名 Π。

7.1.1 生成随机数

签名实体生成一个随机数，它是一个整数 K，其中 $0<K<N$。依赖于该机制，可以在这个生成过程中存在某些附加的约束。这一步的输出为 K，签名实体将其秘密保存。

7.1.2 计算预签名

这一步的输入为随机数 K。其输出为预签名 Π，计算公式如下：

$$\Pi = K^V \bmod N$$

7.2 准备消息

由消息 M 导出两个数据字段 M_1 和 M_2，如 GB/T 17902.1—1999 的 8.2 中所描述。

准备消息的过程应满足下列两个条件之一：

a) 整个消息 M 可以由 M_1 和 M_2 重新构造出；

b) 要找出两个不同的消息 M 和 M'，使得导出的对(M_1，M_2)与(M_1'，M_2')相等，在计算上是不可行的。

第一种情形的典型情况是，$M_1 = M$（当 M_2 为空时），或 $M_2 = M$（当 M_1 为空时），或 $M_1 = M_2 = M$。在第二种情形中，M_1 或 M_2 或两者都是 M 的散列值。

7.3 计算证据

使用一个抗碰撞散列函数(见图 1)计算出 M_1 的散列值 H，它是确定性的证据。

随机化证据 R 定义为一个可选字段的串接，它可用于标识散列函数和填充方法，并使用抗碰撞散列函数来计算。如果 M_1 不为空的话，R 依赖于预签名 Π，并且可选择地依赖于 M_1(见图 2)。

注：除非散列函数是由签名机制或域参数唯一确定的，散列函数标识符应包含在散列权标中。

7.4 计算签名

在基于身份的签名机制中一个签名的计算由下列步骤组成：

a) 计算签名的第一部分；

b) 计算赋值；

c) 计算签名的第二部分。

7.4.1 计算签名的第一部分

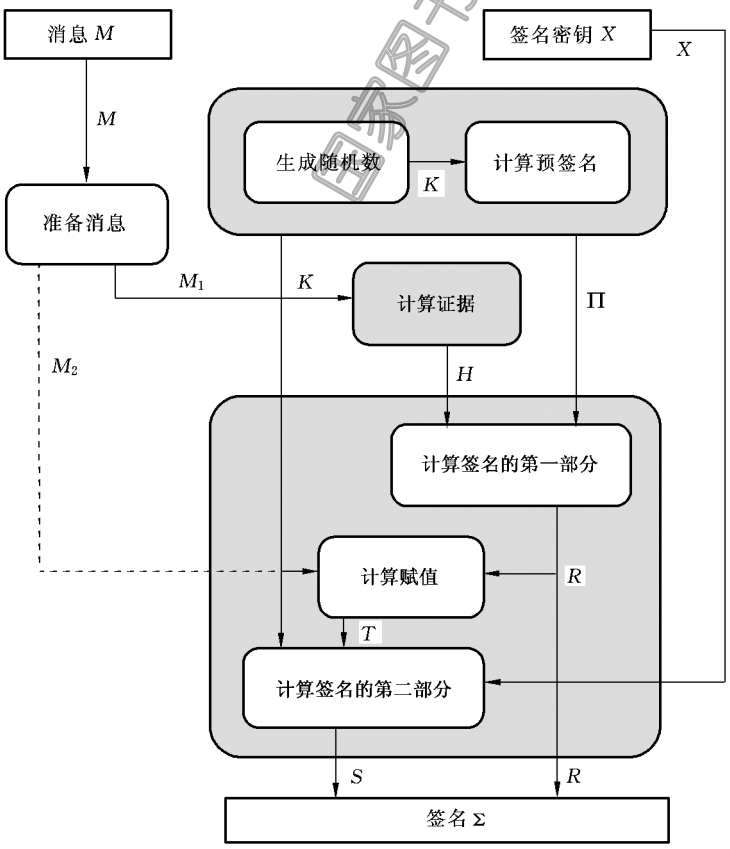

图 1 带确定性证据的签名过程

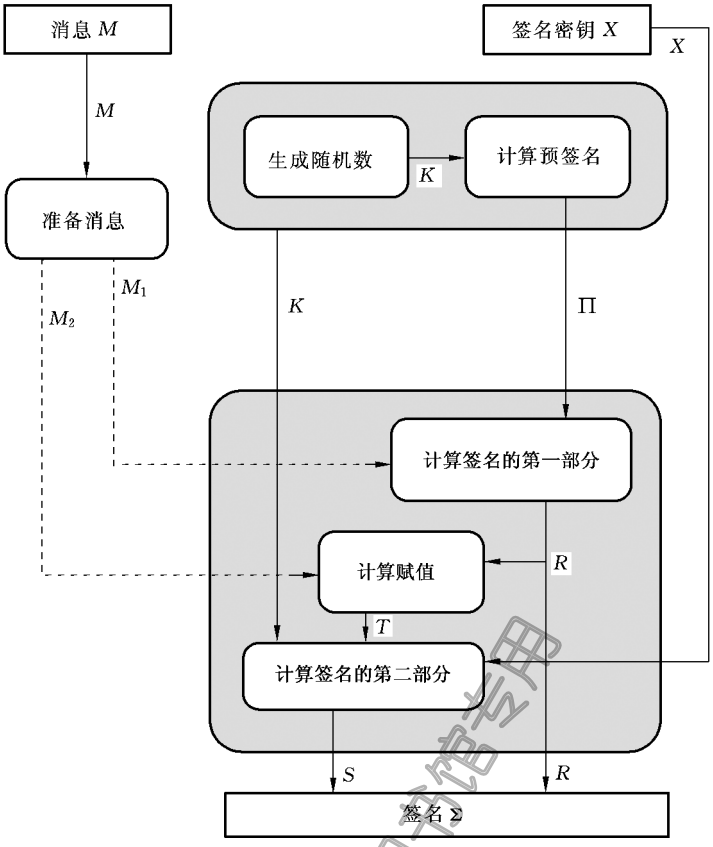

图 2　带随机化证据的签名过程

如果证据是确定性的，计算出签名的第一部分 R，它是 H 和 Π 的函数。这个函数在下列方式下是可逆的：

给定 Π 和 R，散列权标 H 可以被检索（见图 1）。

如果证据是随机化的，则它是签名的第一部分 R，并且不需要做进一步的计算（见图 2）。

注：域参数必须包含一个各方同意的转换一个比特串为一个正整数的方法，以使这一步得以实现。

7.4.2　计算赋值

赋值 T 是一个正整数，它是作为签名的第一部分的函数计算出来。如果 M_2 不为空，该赋值依赖于消息的第二部分 M_2。

计算一对 (R,T) 的方法需要满足如下条件：

要找出具有相同结果对 (R,T) 的任何两对 (M,Π) 和 (M',Π')，在计算上是不可行的。

7.4.3　计算签名的第二部分

基于身份的签名机制的签名函数具有如下形式：

$$S = K \cdot X^{T'} \bmod N$$

其中 K 是 7.1.1 中计算出的随机数，T 是 7.4.2 中计算出的赋值，X 是签名密钥，而 N 为域模数。签名函数的输出为签名的第二部分 S。

8　验证过程

验证过程由下列步骤组成：

a) 准备消息；

b) 检索证据；

c) 计算验证函数；

d) 验证证据。

在验证过程的开始阶段，验证方必须具有如下可用的数据项值：

a) 域参数 N 和 V；

b) 签名者的验证密钥 Y；

c) 消息 M；

d) 签名 $\Sigma=(R,S)$；

e) 可选文本(取自于附录)。

签名的成功验证意味着该签名是采用对应于验证密钥 Y 的签名密钥 X 来创建的。

8.1 准备消息

这个过程等同于 7.2。

8.2 检索证据

如果证据是确定性的，则本过程在 7.3 中描述。验证过程在图 3 中描述。

如果证据是随机性的，验证过程如图 4 中所描述。检索证据是签名的第一部分 R。

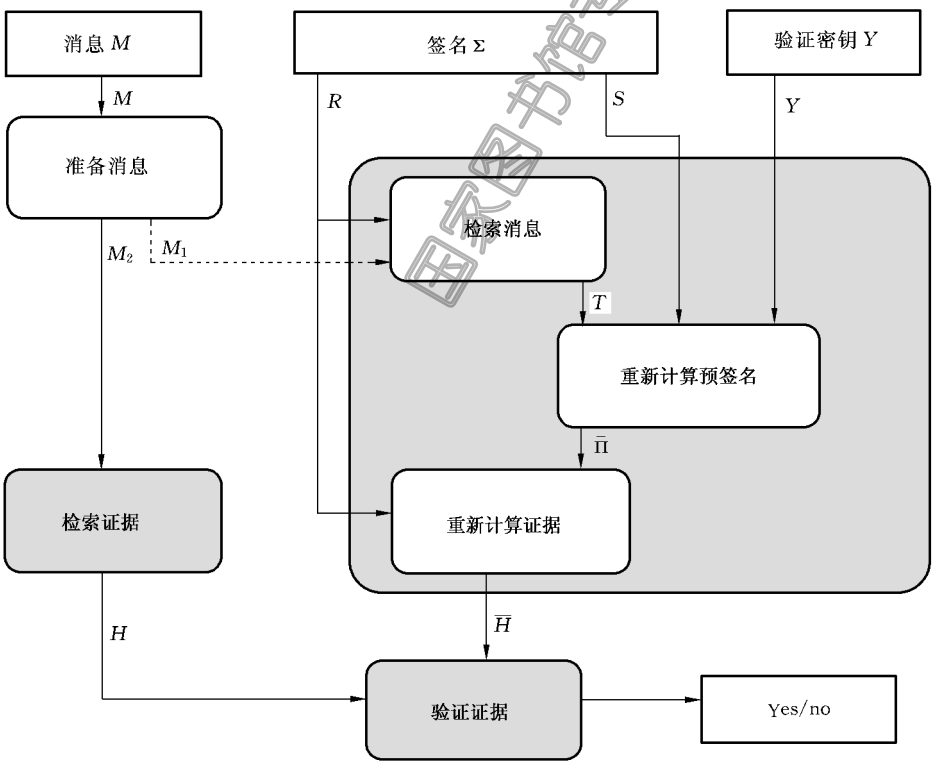

图 3 带确定性证据的验证过程

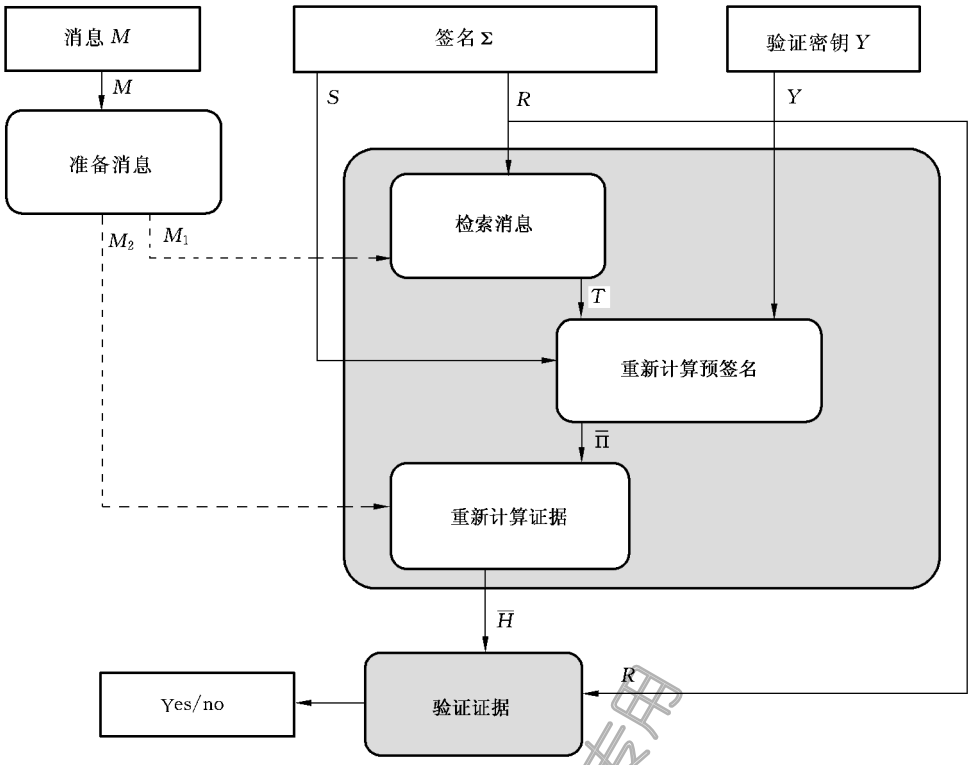

图 4　带随机化证据的验证过程

8.3　计算验证函数

基于身份的签名机制的验证函数的计算由下列步骤组成：

a)　检索赋值；

b)　重新计算预签名；

c)　重新计算证据。

8.3.1　检索赋值

该步骤等同于 7.4.2。如果 M_2 不为空，验证者计算赋值 T，它是在 8.2 中检索出的值 R 和消息 M_2 的函数。赋值 T 是个正整数。

8.3.2　重计算预签名

验证者通过使用以下公式生成预签名的重新计算值 $\bar{\Pi}$：

$$\bar{\Pi} = Y^T \cdot S^V \bmod N$$

其中 Y 为验证密钥，N 为域模数，V 为域验证指数，T 为在 8.3.1 中检索出的赋值而 S 是签名的第二部分。

8.3.3　重计算证据

如果证据是确定性的，它是 M_1 的散列值 H。验证者使用 R 和 $\bar{\Pi}$ 计算出重新计算值 $\bar{H}$。

如果证据是随机性的，其计算等同于 7.3，其中输入数据为 Π 的重新计算值 $\bar{\Pi}$ 和 M 的重新计算值 M_1。输出数据是重计算证据 $\bar{R}$。

8.4　验证证据

该步骤对 8.2 中检索出来的证据数据值和 8.3.3 中重新计算出来的证据数据值进行比较。如果这两个证据数据值相等则验证成功。

9　Guillou-Quisquater 签名机制

Guillou 和 Quisquater 的基于身份的数字签名方案的密钥生成过程、签名过程和验证过程在

GB/T 17902本部分中的第6章和第8章及图2和图4中描述。

现在对下面的过程和函数进行更详细地说明：

a) 公钥导出函数；

b) 准备消息；

c) 计算证据；

d) 计算签名的第一部分；

e) 计算赋值。

9.1 公钥导出函数

公钥导出函数是GB 15851—1995中描述的冗余发生函数。签名实体的标识数据I，或它的散列权标在GB 15851—1995的5.1到5.4中描述的过程中被作为输入消息。实体的公开密钥$Y = y(I)$设置为这个过程的输出数据，它在GB 15851—1995中被称为中间整数。

9.2 准备消息

在Guillou-Quisquater机制中，$M_1 = M$并且M_2为空。

9.3 计算证据

在Guillou-Quisquater机制中，证据R被计算出来，它是数据$\Pi \| M$的散列权标，其中Π是预签名：

$$R = H(\Pi \| M)$$

9.4 计算签名的第一部分

在Guillou-Quisquater机制中，证据R构成签名的第一部分。

9.5 计算赋值

赋值T等于R，其表示为一个正整数。

注：域参数必须包含一个各方同意的转换一个比特串为一个正整数的方法，以使这步得以实现。

10 带短赋值的基于身份的签名

在本章中规定了Guillou-Quisquater方案的一个变型，它适用于带低速处理器或低速输入/输出接口的设备上的实现。附加的散列信息用于减少中间参数的长度。多达三个不同的散列函数可以被用在证据和赋值的计算中。它们不需抗碰撞，其选择必须满足这样的条件，即要找出具有相同的(R,T)的两个不同的消息在计算上是不可行的。

这个机制在下列过程中，不同于Guillou-Quisquater方案：

a) 准备消息；

b) 计算证据；

c) 计算签名的第一部分；

d) 计算赋值。

10.1 准备消息

在这个变型方案中，消息输入的两个部分M_1和M_2设置为消息M的散列权标H。

10.2 计算证据

证据R是通过计算数据$H_1 \| H$的散列权标H_2得到的。其中H是在10.1中计算出来的，它是预签名Π的散列权标。

注：预计算散列权标H_1可以被更有效地存储。

10.3 计算赋值

赋值T是由计算数据$H \| R$的散列权标H_3得到的。

11 带消息散列码检索的基于身份的签名

Guillou-Quisquater方案中的这个变型的优点，是散列函数可以与其余的验证步骤一起并行计算。

基于身份的数字签名机制的签名过程和验证过程中描述在图 1 和图 3 中。这个机制在下列过程方面与 Guillou-Quisquater 方案不同：

a) 计算证据；

b) 计算签名的第一部分。

另外，它假设由签名者生成的随机数 K 满足 $K \neq 0 \bmod P$ 和 $K \neq 0 \bmod Q$，否则 N 的因子分解将被找出。因此，还可以假设预签名 Π 不是 P 或 Q 的倍数。

11.1 计算证据

使用一个抗碰撞散列函数将证据计算出来，它是消息 M 的散列权标 H。

11.2 计算签名的第一部分

计算出签名的第一部分 R 如下：

$$R = \Pi \cdot H \bmod N$$

给定一个预签名的重新计算值 $\overline{\Pi}$，验证者通过如下计算得到证据的重新计算数值 $\overline{H}$：

$$\overline{H} = \overline{\Pi}^{-1} R \bmod N$$

附 录 A
（资料性附录）
数 值 例 子

使用十六进制记法。

A.1 密钥生成过程的数值例子

A.1.1 生成域参数

域模数 N 是两个不同的素数 P 和 Q 的乘积。这个例子中，P 和 Q 是 512 比特的素数，因此 N 是 1024 比特的数。

P = FFFFFFFF EA2DE66E D3B1B7E9 61B75DFC D9FAE2FF
A07A2345 9B7956FB 1B9B16D7 E1B6D59B BDF45B85
3CBF08EA 3BC7A1BD 541CB3A8 80E02E43 87CA7DEF
50948E87

Q = FFFFFFFF E275B7F4 98A3811D E906ACF7 BFEB5CD6
A445AF09 D7906DE1 97CC2CCD 87614718 8C7C084F
CE9231CA B7CFA113 13C3DDCF F1B70A54 84494467
8FCEF193

$N = PQ$ =
FFFFFFFF CCA39E63 6ED9CF52 950C23A0 38AE0291
012B984A 964FFBBD 99E9DACB 91400431 0C5DD264
B1873126 44A725C5 D5BC73F4 97CFD100 89FD1342
656026BE 3FB583FE B134FF43 6957A1E1 D975B5BE
DF1A9570 4C81A337 F06E5F9F 9388A7AC 5ABFD5CF
0356D91A 9861C69F E50509C2 323E5270 F2015FBD
C08AA2C0 391CEE85

域验证指数是个奇整数，它与 $P-1$ 和 $Q-1$ 互素。在这个例子中，V 为 $2^{79}+1$，因此 V 的长度是 80 比特。

V = 8000 00000000 00000001

密钥生成指数 D 是正整数，使得 $DV-1$ 是 lcm $(P-1, Q-1)$ 的倍数。

lcm $(P-1, Q-1)$ =
7FFFFFFF E651CF31 B76CE7A9 4A8611D0 1C570148
8095CC25 4B27FDDE CCF4ED65 C8A00218 862EE932
58C39893 225392E2 EADE39FA 4BE7E880 44FE89A1
32B0135E 1FDAC1FF 7248B06F FE81346D 475BD565
229A2ACD 03E0E874 3EB24D61 7010B203 78D3DC8D
5D733AA2 C68845F5 78B6E378 E52EE07C 3FB51392
DA3B7034 AC5CB736

D = 1BC6C0ED 36435CBF A89C7A35 50CE3D54 C6ABC9F5
EE5E75C9 E458AADA 6178CB20 C7339C4E F30413A6
586DA8B6 45A72BDF 291C9218 F0CA83EF A4234FAD
8394B2BF 8F4A0EF9 61E098FC 2CC5AFAA 46CCC821

0427D3EE 3461AF0B 46895311 E1DAD21F 35217CBC
4FD1A5B9 62E01B8B 967F97E2 41ECF56E DBF85278
EC058601 17D9A7B7

A.1.2 生成验证密钥和签名密钥

验证密钥 Y 是以标识数据 I 作为输入的公钥导出函数的输出，并且不是 P 或 Q 的倍数。这个例子使用一个小于 N 的 1024 比特的数 Y。

$Y =$ C50ECCC9 64443B0A 1C974F40 1C94E500 FA8214FC
9B1B5EC5 2AA1201A 001EA009 FE90D01D F32C6B43
323F0812 42ABE843 09F926BB 9338A841 5DEF2EF6
E709E3BD 515B5D86 C3ED4B7F C15FA876 26E8E9C7
0E557D5B A8E96D7C B55FBF41 37F601FF 47B7CCCB
6BED4407 6F8E9805 42E37105 522E7184 42A717DF
E89A6B62 7B6E60B7

可信机构计算私有签名密钥如下：

$X = Y^{-D} \bmod N =$
A763FA4 3895CFDD D80627A6 A8271250 97C184E5
10F0075C 48FCB0E7 F2885275 AAA32829 C08CF352
0F42F6FD C296DCE1 F50FBDED D5C33C7C 63298C4F
26C2CDEE 11D927BA C6EC4A6A C022C063 1F30E880
07452397 7F3ACA8C 422E2461 3B7F3BB0 E61D04B8
0670A128 0ED7C8C1 A72D4B1C C566381B 0665F83B
70FD7158 0B7A6EEC

A.2 在第 9 章中描述的 Guillou-quisquater 签名机制的数值例子

在这个例子中，所使用的散列函数是固定的，并且为域内的所有实体已知，因此不需要散列标识符。通过使用 SHA-1 生成散列权标。

域参数、私有签名密钥和验证密钥与第 A.1 章相同。

A.2.1 签名过程

A.2.1.1 生成预签名

签名实体生成一个随机数，它是一个随机性的或伪随机性的整数 K，其中 $0<K<N$。本例使用一个 1024 比特的随机数 K。

$K =$ B2045A19 83150F5B B04CB524 2B566A37 6779F416
5C7F1673 029C1BFC 05A84C60 E401897A CEAD9DE5
C7D8108B 95943332 FF6B20D3 004CCD40 36BDBB7E
10DA755E B03720F0 5A0CDC53 66EB4374 BE091A80
6D3399190 D2ADE1CD 9E11EF4E A5FF6969 3D0EB942
BFCC333D F5FDD599 1D3B78A7 4868B0C6 381AEA61
C24F3E05 2D8D9FFA

预签名 Π 计算如下：

$\Pi = K^{V} \bmod N =$
15B35BB7 0DDD7ED8 0FAD7EDE A80F828E 46B3F86D
4EFB7E84 58562B6D 6F1885D0 A02FD892 8838C128
B53EE703 FAC96534 6C18A714 17D12FD7 211C3956

B6AD1A15 F4399EB0 CC065E8F CC0039F3 E13A8EFE
5FB384CC D0190FA9 DC995DEC BB07947F 72124D7B
9044433C 7A416B62 99297387 0174FB6A 94A8FA8B
1CBD7E0A 2780DFD8

A.2.1.2 准备消息

在本例中,消息为如下文本段:

M:This is a test message!(这是一个测试消息!)

A.2.1.3 计算证据

在本例中,证据形成签名的第一部分:

$R = \Pi \| M$ 的散列权标 =

42AF9098 F1596D17 EAC6EA31 36E7D9F3 30F8E9FB

A.2.1.4 计算赋值

赋值 T 等于 R,表示为一个正整数。

$T=R$

A.2.1.5 计算签名的第二部分

签名的第二部分 S 计算如下:

$S=K \cdot X^{T} \bmod N =$

B7FAF642 59F99E7A B32029A6 1FDEE247 F068853F
80C6FF87 9FB07983 A61C047B 34CEE1C8 F69FF97B
020F0B6C F37E4A85 05EAEF00 01E825E5 8B8F3438
0C332BF5 B3D47E5F C654747F E1289D61 061F124B
C19EBB38 D466970C 39CEC404 703FB359 A0019692
F968F760 72F2D6F5 0CD75C73 1D9EEE7F EFFB6F98
E80BD095 5D00C051

签名 Σ 被设置为(R,S)。

A.2.2 验证过程

验证者拥有下列可用数据项值:

$N,V,Y,M,\Sigma = (R,S)$

A.2.2.1 检索赋值

赋值 T 等于签名的第一部分 R:

$T=R$

A.2.2.2 重新计算预签名

验证者生成预签名的一个重新计算数值 $\bar{\Pi}= Y^{T} \cdot S^{V} \bmod N$:

$\bar{\Pi}=Y^{T} \cdot S^{V} \bmod N$

注:如果$(Y^{T} \bmod N)$和$(S^{V} \bmod N)$是分别计算的,它们则给定如下:

$Y^{T} \bmod N =$

3C18CAA0 339ED6CA 6C80AF2C 201481C1 74054C22
A8314537 0B8C6DF1 92766BC4 C3FD8C14 76EDD630
E02E33E8 4F557C86 A51FE7B5 9769CF40 98E34D29
111E9BF0 82825727 85B047F3 82F51DB4 51558F87
FE5FAA6F C1C45803 77AC051D 85A094BD 65472145
4518DECB EC2B58FF 791DC06F D202E815 1A39ACA5
8D5C60D9 419695E7

$S^V \bmod N =$

A8E6DDCF	74D3C94D	23F5FDE0	820431B4	FA51F3D9
150D5CDE	DB0692A0	735FD729	BCAC0825	7CA19BA1
565F03BA	7849B538	B9BF1797	5473FC28	299F0F3C
04C3EE35	E31BFA2C	A17F1781	0488F988	05439C3E
7B8BE70C	F027401D	786DAC48	AE2507F5	30EC01C7
1090504D	9BD82673	CD472405	8EF34697	21F90D3F
1E2DD798	50AC0090			

A.2.2.3 重新计算证据并验证证据

证据被重新计算出来，它是散列权标 $\bar{\Pi} = Y^T \cdot S^V \bmod N \parallel M = R$。

重新计算出来的证据 $\bar{\Pi} = Y^T \cdot S^V \bmod N$ 与检索出来的证据 R 相同。

A.3 在第 10 章中描述的带有短赋值的基于身份的签名的数值例子

在这个例子中，所使用的散列函数是固定的，并且为域内的所有实体已知，因此不需要散列标识符。通过使用 SHA-1 生成散列权标 H，H_1 和 H_2。另外一个散列权标 H_3 是通过下列函数 h 计算出来的，具有两个 160 比特的输入数据，它产生一个 80 比特的输出：

$h(u \parallel v) = \{(u_1 \oplus u_2) + (v_1 \oplus v_2)\} \bmod 2^{80}$

其中 $u = u_1 \cdot 2^{80} + u_2$ 和 $v = v_1 \cdot 2^{80} + v_2$，$\oplus$ 是比特与比特间的异或操作。

域参数、私有签名密钥和验证密钥与第 A.1 章中相同。

A.3.1 签名过程

A.3.1.1 生成预签名

签名实体生成一个随机数，它是一个随机性的或伪随机性的整数 K，其中 $0 < K < N$。本例使用一个 1024 比特的随机数 K，而相应的预签名 Π 在 A.2.1.1 中给定。

H_1 是预先计算出来的并且存储在一个存储器中用于更有效地计算证据。

$H_1 =$ 预签名 Π 的散列权标 $=$

7B2B1483　46B8D0B0　F42C762B　FE93D691　C9BC3841

A.3.1.2 准备消息

在本例中，消息与 A.2.1.2 中的消息相同。

$H =$ 消息 M 的散列权标 $=$

A9D66D4B　652597FB　32DD1092　E7C9CDE1　8F0C7FBC

A.3.1.3 计算证据和签名的第一部分

在本例中，证据形成签名的第一部分 R。

$R = H_2 =$ 散列权标 $H_1 \parallel H =$

D6149DF3　DA7E60EA　817CAADE　D2F9F785　8800903F

A.3.1.4 计算赋值

赋值 T 是通过计算数据 $R \parallel H$ 的散列权标 H_3 来得到的。

$T = h(H \parallel R) =$ 360E　D98CD6C0　01E15EA4

A.3.1.5 计算签名的第二部分

签名的第二部分 S 计算如下：

$S = K \cdot X^{r} \bmod N =$

FA17252E	94A3D992	C5CE7953	D0939F9E	45A4AE8A
A9F2B89F	1345528F	658959C9	B5B16E92	9131A3A0
BFA8490C	E4652452	90ECCFDF	73E67220	7D34AFA0
F0B1C381	EDE82B2C	2B2D520E	4E91EB67	98922DF5

04B1883E D7CD9AC3 1E3E4E99 FF2E57F9 ACE6F5E6
475A029E 1821081F 02DB73ED 6640A148 EF281E9B
63E3B7E1 DA0E9AA9

签名 Σ 被设置为(R,S)。

A.3.2 验证过程

验证者拥有下列数据项值可用：

$N,V,Y,M,\Sigma=(R,S)$

A.3.2.1 准备消息

$H = M$ 的散列权标

A.3.2.2 检索赋值

赋值 T 是通过计算数据 $R \parallel H$ 的散列权标 H_3 得到的。

$T = h(H \parallel R)$

A.3.2.3 重新计算预签名

验证者生成预签名的重新计算值 $\bar{\Pi} = Y^T \cdot S^V \bmod N$。

如果$(Y^T \bmod N)$和$(S^V \bmod N)$是分别计算的，它们则给定如下：

$Y^T \bmod N =$

390AB7EB 69A80EE0 D382FFA6 6D6557B3 D17D68A4
CE5D67F1 FCF76F27 D660941A 90941033 D3269612
DC989EA4 78257463 8C98ADEA F05DAC71 87A000BE
C7E20991 9B0598D1 47B22923 6C45DF67 358E4DEF
900DC37C 8812CD7A D827A550 3703BC3C DE843E4E
B12C7C16 DBB7AD5B 98F4F0D9 D4918CA5 32060A48
E58A9445 437328E9

$S^V \bmod N =$

3A65050A 97021E9A 8A6E825A 26EE068D EF06D046
0617D9FE 3C577159 06C15DD0 2EBF8707 CC681798
373B4E16 9A12A839 04B7CBAF D578BC4E 7D0FD50D
0C7E549C CECC18BA A198682F CB46844A E5C904E4
A9003E35 A0035D96 B170F5A7 E1C5B285 4F66C7BD
8132EB24 A9F3EDF1 10EC1C02 377B0963 ACEFBB7B
8A48243B F771D47F

A.3.2.4 重新计算证据并验证证据

证据被重新计算出来，它是 $\bar{\Pi} = Y^T \cdot S^V \bmod N$。

$\bar{\Pi} = Y^T \cdot S^V \bmod N = R$

重新计算出来的证据与检索出来的证据 R 相同。

A.4 在第 11 章中描述的给出消息散列代码检索的基于身份的签名的数值例子

在这个例子中，所使用的散列函数是固定的，并且为域内的所有实体已知，因此不需要散列标识符。通过使用 SHA-1 生成散列权标。

域参数、私有签名密钥和验证密钥与第 A.1 章中相同。

A.4.1 签名过程

A.4.1.1 生成预签名

签名实体生成一个随机数，它是一个随机性的或伪随机性的整数 K，其中 $0 < K < N$。本例使用

一个 1024 比特的随机数 K，而相应的预签名 Π 在 A.2.1.1 中给定。

A.4.1.2 准备消息

在本例中，消息与 A.2.1.2 中的消息相同，并且它的散列权标 H 与 A.3.1.2 中的相同。

$H = M$ 的散列权标

A.4.1.3 计算证据和签名的第一部分

在本例中，证据形成签名的第一部分 R。

$R = \Pi \cdot H \bmod N =$

425DCEDD	1D408F3F	6633CEFE	225B92DE	920BE1AF
BD2CE776	446410E7	A08527BC	5ADA0DDD	13C8B371
053FD47C	69BD86FA	1114EE0C	698B7E10	1662529C
9B53662D	64BCD974	4DCF5276	BF576154	407AB43F
85BCC21B	2075B492	142A5724	464A6E30	21777BD0
C5BBC02F	7B93F42C	07916DA7	1BD4D276	81D21D5A
58F5A5AC	D8C3A8EA			

A.4.1.4 计算赋值

在本例中，赋值 T 等于 R。

$T=R$

A.4.1.5 计算签名的第二部分

签名的第二部分 S 计算如下：

$S=K \cdot X^T \bmod N =$

3205E60E	84EB7AC6	28866B9D	2CA2A080	D182BF95
27FE80D3	EC2EF3F9	F27F0FF3	4DCEC086	BDB0E072
08EDE468	E5C2F36C	25452213	A3AD3731	3A518CF9
1DE84C4D	25AA2AB9	84D76885	CA9B8A2C	DB388F1D
139AA00F	15340501	D6B7F867	D5313E2B	60F64E34
F7650FED	23C032EB	DDB82BD0	A5AB49FA	CC5F5AAB
AB17F14A	180E6A76			

签名 Σ 被设置为(R,S)。

A.4.2 验证过程

验证者拥有下列数据项值可用：

$N,V,Y,M,\Sigma =(R,S)$

A.4.2.1 准备消息

$H = M$ 的散列权标

A.4.2.2 检索赋值

赋值 T 等于 R。

$T=R$

A.4.2.3 重计算预签名

验证者生成一个预签名的重新计算值 $\bar{\Pi}=Y^T \cdot S^V \bmod N$。

如果$(Y^T \bmod N)$和$(S^V \bmod N)$是分别计算的，它们则给定如下：

$Y^T \bmod N =$

E7D8C6CC	45A8B29F	200F5C52	148824E0	771624FC
DE0FDD70	BC83BE2E	22AAFE20	C0233C14	E6E4E8E2
BEEE7C5C	022FD464	92D0A055	5A0D7782	7AFBEFB4

6D8085FD AF8A27C1 775285B7 DD7894F4 F05DAF37
BFCF170C B3638CE7 A796D639 B296D8D9 09B8DE8F
8AC5A98F 3BBDA623 3BB66C95 5127A854 4DC7206B
ED0EB25D 0AB17CDC

$S^V \bmod N =$

F673DE2 802FA537 DE498937 F969AA84 9F9CE1F0
708BDEB3 D389CD31 2F5BDBD1 C036BDD4 449D4FB4
00AD359F 02B997A6 273B23CF 1F414C1F C27B7556
74D64887 1B2A3185 12E098D9 2ED260C8 CC511579
886C4EA2 2D638A5A B727D984 B6F1C2E4 501761BA
74066387 CB0DBA34 A9822A32 E1761CE8 4622E52D
71AB16D9 267B748E

A.4.2.4 重新计算证据并验证证据

证据被重新计算如下：

$\bar{\Pi} = Y^T \cdot S^V \bmod N = \bar{\Pi} = Y^T \cdot S^V \bmod N \cdot H \bmod N = R$

重新计算出来的证据 $\bar{\Pi} = Y^T \cdot S^V \bmod N$ 与检索出来的证据 R 相同。

给定预签名的重新计算值 $\bar{\Pi}$，验证者通过下列计算得到证据的重新计算值 $\bar{H}$。

$\bar{H} = \bar{\Pi}^{-1} R \bmod N =$

43F85D2C D63CD833 74D74837 591623A2 26734169
5BF58243 070B14C7 8B93E68F 4FFF6D7E FA150C1D
052ECB5F D743721D 2481469B C1375059 3A05C239
98E9FA85 3EE7634E 013726BF 8A04FDD3 A9DEA82D
2B93F990 55E842BE A40DCB27 0E74F8BC 1A243E46
A274BD40 7B5B7A40 45B5F936 B30CAA2E 7C4F1F72
711EA20E E3CEEFD2

$\bar{H} = \bar{\Pi}^{-1} R \bmod N = \bar{\Pi}^{-1} R \bmod \cdot \bar{\Pi} = Y^T \cdot S^V \bmod N$

附 录 B
（资料性附录）
专 利 信 息

GB/T 17902 的本部分在制定中等同采用了国际标准 ISO/IEC 14888-2:1999 附录 B 的专利信息。

关于专利的使用遵照国家有关规定。

国际标准的有关专利信息如下：

在该国际标准(ISO/IEC 14888 第 2 部分)的准备期间，收集了该国际标准的应用所需要的相关专利信息。相关专利如下表所示。但是，关于专利的有效性或范围，ISO/IEC 没有给出权威的或全面的信息。

这些注册专利的持有者已经声明，如果寻求授权许可的使用方同意付费，将在适当的条件下给予许可，使其能够应用本部分。

进一步的信息可从专利持有者处得到。

专业范围	发明者	专利号	发布日期	联系地址
Fiat-Shamir 身份鉴别	Shamir-Fiat	US 4,748,668	1988-05-31	News Difital Systems Ltd. Stoneham Rectory Stoneham Lane Eastleigh Hampshire SO50 9NW, UK
GQ 身份鉴别	Guillou-quisquator	US 5,140,634	1992-08-18	CCETT Patent and IPR Office BP 59 4 Rue du Clos Courtel F-35512 Cesson Sevigne France
		EP 0.311.470	1992-12-16	Philips International B. V. Corporate Patents and Trademarks P. O. Box 220 56000 AE Eindhoven The Nether lands

参 考 文 献

[1] Guillou Louis C., Quisquater Jean-Jacques, "A practical zero-knowledge protocol fitted to security microprocessor minimizing both transmission and memory", Advances in Cryptology-Eurocrypt '88, Christoph G. Gunther (Ed.), Lecture Notes in Computer Science 330, Springer-Verlag, 1988, pp. 123-128.

[2] M. Girault & J. Stern, "On the length of cryptographic hash-values used in identification schemes," Advances in Cryptology-Crypto'94, Y. Desmedt (Ed.), Lecture Notes in Computer Science 330, Springer-Verlag, 1994, pp. 202-215.

GB/T 17902.2-2005

*

书号:155066·1-23069