(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 102263638 B
(45)授权公告日 2016.08.10

(21)申请号 201110145023.8

(22)申请日 2011.05.24

(30)优先权数据
2010-125026 2010.05.31 JP
2010-224753 2010.10.04 JP
2011-026401 2011.02.09 JP

(73)专利权人 索尼公司
地址 日本东京都

(72)发明人 作本纮一 白井太三 樋渡玄良

(74)专利代理机构 北京集佳知识产权代理有限公司 11227
代理人 陈炜

(51)Int.Cl.
H04L 9/32(2006.01)
H04L 9/30(2006.01)

(56)对比文件
US 2007/0110232 A1,2007.05.17,
CN 101277186 A,2008.10.01,
JP 特表 2009-533705 A,2009.09.17,

审查员 张倩

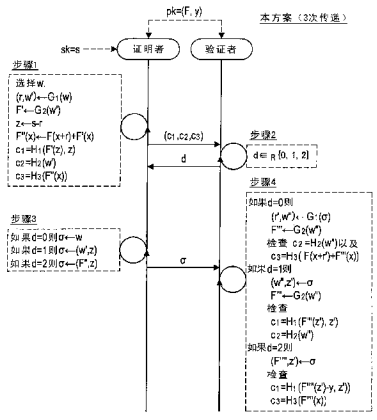
权利要求书6页 说明书47页 附图27页

(54)发明名称

认证设备、认证方法和签名生成设备

(57)摘要

本发明公开了认证设备、认证方法和签名生成设备。提供了一种认证设备，包括：密钥设置单元，用于将 $s \in K^n$ 设置为私钥并且将环K上的多次多项式 $f_i(x_1,\cdots,x_n)$ $(i = 1$ 至 $m)$ 以及 $y_i = f_i(s)$ 设置为公钥；消息发送单元，用于将消息c发送给验证者；验证模式接收单元，用于接收与验证者针对一个消息c从k($k \geq 3$)个验证模式中选择的一个验证模式有关的信息；以及响应发送单元，用于向验证者发送k种响应信息中的与由验证模式接收单元接收到的验证模式有关的信息所对应的响应信息，其中，响应信息是这样的信息，该信息使得在通过利用k种响应信息执行的针对消息c的所有k个验证模式都成功的情况中能够计算出私钥s。



CN 102263638 B

1.一种认证设备，包括：

密钥设置单元，用于将$s \in K^n$设置为私钥并且将环K上的多次多项式$f_i(x_1,\cdots,x_n)$设置为公钥或系统参数，并将$y_i = f_i(s)$设置为所述公钥，i＝1至m，其中，K^n表示K的n次直积；

消息发送单元，用于将消息c发送给验证者，其中，所述认证设备试图向所述验证者证明合法性；

验证模式接收单元，用于接收由所述验证者从针对一个消息c的多个验证模式中选择的一个验证模式的信息；以及

响应发送单元，用于向所述验证者发送所述多个验证模式的响应信息中的与由所述验证模式接收单元接收到的验证模式的信息相对应的响应信息。

2.根据权利要求1所述的认证设备，其中，所述响应信息是使得在如下情况中能够计算出所述私钥s的信息：通过利用所述多个验证模式的响应信息执行的针对消息c的所述多个验证模式的全部已成功。

3.根据权利要求2所述的认证设备，

其中，如果在执行由所述消息发送单元发送一个或多个消息c的第一步骤、由所述验证模式接收单元针对每个消息c从所述验证者接收验证模式的信息的第二步骤、以及由所述响应发送单元针对每个验证模式的信息发送响应信息的第三步骤时，所述验证者利用所有响应信息都成功地执行了验证，则认证成功。

4.根据权利要求3所述的认证设备，

其中，执行由所述消息发送单元发送一个或多个消息c的第一步骤、由所述验证模式接收单元针对每个消息c从所述验证者接收验证模式的信息的第二步骤、以及由所述响应发送单元针对每个验证模式的信息发送响应信息的第三步骤的处理被重复，并且

其中，如果当将所述第一步骤至第三步骤执行了预定次数时，所述验证者利用所有响应信息每次都成功地执行了验证，则所述认证成功。

5.根据权利要求3或4所述的认证设备，

其中，在所述消息$c=(c_1,\cdots,c_m)$的情况中，

所述消息发送单元通过利用单向函数H来计算新的消息$c' = H(c)$，并且将所述消息c'发送给所述验证者，并且

所述响应发送单元将所述验证者即使利用所述响应信息也不能恢复出来的所述消息c的元素与所述响应信息一起发送。

6.根据权利要求5所述的认证设备，其中，所述单向函数是哈希函数或承诺函数。

7.根据权利要求1所述的认证设备，其中，所述响应信息是使用基于所述私钥s和向量$r \in K^n$算出的$z \in K^n$、基于所述向量r和$t \in K^n$算出的$t' \in K^n$和基于将所述向量r代入所述多次多项式f_i的$f_i(r)$和$e_i \in K$算出的$e_i' \in K$而计算出的。

8.根据权利要求1所述的认证设备，其中，所述多次多项式f_i是二次多项式。

9.根据权利要求1所述的认证设备，

其中，所述m和所述n具有$m < n$的关系。

10.根据权利要求9所述的认证设备，

其中，所述m和所述n具有$2^{m-n} << 1$的关系。

11.一种认证设备，包括：

消息接收单元，用于从证明者接收消息c；

验证模式选择单元，用于从针对一个消息c的多个验证模式中选择一个验证模式；

验证模式发送单元，用于将由所述验证模式选择单元选择的验证模式的信息发送给所述证明者；

响应接收单元，用于从所述证明者接收所述多个验证模式的响应信息中的与由所述验证模式发送单元发送的验证模式的信息相对应的响应信息；以及

验证单元，用于利用由所述消息接收单元接收的所述消息c和由所述响应接收单元接收的所述响应信息来验证所述证明者的合法性，

其中，$s \in K^n$被设置为私钥，并且环K上的多次多项式$f_i(x_1,\cdots,x_n)$被设置为公钥或系统参数，并且$y_i = f_i(s)$被设置为所述公钥，i＝1至m，其中，K^n表示K的n次直积。

12.根据权利要求11所述的认证设备，其中，所述响应信息是使得在如下情况中能够计算出所述私钥s的信息：通过利用所述多个验证模式的响应信息执行的针对消息c的所述多个验证模式的全部已成功。

13.根据权利要求11所述的认证设备，其中，所述响应信息是使用基于所述私钥s和向量$r \in K^n$算出的$z \in K^n$、基于所述向量r和$t \in K^n$算出的$t' \in K^n$和基于将所述向量r代入所述多次多项式f_i的$f_i(r)$和$e_i \in K$算出的$e_i' \in K$而计算出的。

14.根据权利要求11所述的认证设备，其中，所述多次多项式f_i是二次多项式。

15.一种认证设备，包括：

密钥设置单元，用于将$s \in K^n$设置为私钥并且将环K上的多次多项式$f_i(x_1,\cdots,x_n)$设置为公钥或系统参数，并将$y_i = f_i(s)$设置为所述公钥，i＝1至m，其中，K^n表示K的n次直积；

消息发送单元，用于将消息c发送给验证者；

答复接收单元，用于从所述验证者接收答复α；

多项式生成单元，用于利用所述答复接收单元接收的所述答复α来生成要用于验证所述消息c的多项式f”；

多项式发送单元，用于将由所述多项式生成单元生成的所述多项式f”发送给所述验证者；

验证模式接收单元，用于接收由所述验证者从针对一个消息c的k个验证模式中选择的一个验证模式的信息，$k \geq 2$；

响应发送单元，用于向所述验证者发送k种响应信息中的与由所述验证模式接收单元接收的验证模式的信息相对应的响应信息。

16.根据权利要求15所述的认证设备，其中，所述响应信息是使得在如下情况中能够计算出所述私钥s的信息：通过利用两种答复α、所述多项式f”以及k种响应信息执行的针对所述消息c的答复与验证模式的总计2k种组合的全部已成功。

17.根据权利要求16所述的认证设备，

其中，如果在执行由所述消息发送单元发送一个或多个消息c的第一步骤、由所述答复接收单元接收针对每个消息c的所述答复α的第二步骤、由所述多项式生成单元利用在所述第二步骤中接收的所述答复α来生成多项式f”并且由所述多项式发送单元发送所述多项式f”的第三步骤、由所述验证模式接收单元针对每个消息c从所述验证者接收验证模式的信息的第四步骤、以及由所述响应发送单元针对每个验证模式的信息发送响应信息的第五步

骤时，如果所述验证者利用所有响应信息都成功地执行了验证，则认证成功。

18.根据权利要求17所述的认证设备，

其中，执行由所述消息发送单元发送一个或多个消息c的第一步骤、由所述答复接收单元接收针对每个消息c的所述答复α的第二步骤、由所述多项式生成单元利用在所述第二步中接收的所述答复α来生成多项式f”并且由所述多项式发送单元发送所述多项式f”的第三步骤、由所述验证模式接收单元针对每个消息c从所述验证者接收验证模式的信息的第四步骤、以及由所述响应发送单元针对每个验证模式的信息发送响应信息的第五步骤的处理被重复，并且

其中，如果当将所述第一步骤至第五步骤执行了预定次数时，所述验证者利用所有响应信息每次都成功地执行了验证，则所述认证成功。

19.根据权利要求15所述的认证设备，其中，所述响应信息是使用基于所述私钥s和向量$r\in K^n$算出的$z\in K^n$、基于所述向量r和$t\in K^n$算出的$t'\in K^n$和基于将所述向量r代入所述多次多项式f_i的$f_i(r)$和$e_i\in K$算出的$e_i'\in K$而计算出的。

20.根据权利要求15所述的认证设备，其中，所述多次多项式f_i是二次多项式。

21.一种认证设备，包括：

消息接收单元，用于从证明者接收消息c；

答复发送单元，用于向所述证明者发送答复α；

多项式接收单元，用于接收多项式f”，所述多项式f”是由所述证明者利用由所述答复发送单元发送的所述答复α生成的并被用于验证所述消息c；

验证模式选择单元，用于从针对一个消息c的k个验证模式中选择一个验证模式，$k\geq 2$；

验证模式发送单元，用于将由所述验证模式选择单元选择的验证模式的信息发送给所述证明者；

响应接收单元，用于从所述证明者接收k种响应信息中的与由所述验证模式发送单元发送的验证模式的信息相对应的响应信息；以及

验证单元，用于利用由所述消息接收单元接收的所述消息c、由所述多项式接收单元接收的所述多项式f”以及由所述响应接收单元接收的所述响应信息，验证所述证明者的合法性，

其中，$s\in K^n$被设置为私钥，并且环K上的多次多项式$f_i(x_1,\cdots,x_n)$被设置为公钥或系统参数，并且$y_i=f_i(s)$被设置为所述公钥，i＝1至m，其中，K^n表示K的n次直积。

22.根据权利要求21所述的认证设备，其中，所述响应信息是使得在如下情况中能够计算出所述私钥s的信息：通过利用两种答复α、所述多项式f”以及k种响应信息执行的针对所述消息c的答复与验证模式的总计2k种组合的全部已成功。

23.根据权利要求21所述的认证设备，其中，所述响应信息是使用基于所述私钥s和向量$r\in K^n$算出的$z\in K^n$、基于所述向量r和$t\in K^n$算出的$t'\in K^n$和基于将所述向量r代入所述多次多项式f_i的$f_i(r)$和$e_i\in K$算出的$e_i'\in K$而计算出的。

24.根据权利要求21所述的认证设备，其中，所述多次多项式f_i是二次多项式。

25.一种认证方法，包括以下步骤：

将$s\in K^n$设置为私钥，并且将环K上的多次多项式$f_i(x_1,\cdots,x_n)$设置为公钥或系统参数，并将$y_i=f_i(s)$设置为所述公钥，i＝1至m，其中，K^n表示K的n次直积；

将消息c发送给验证者；

接收由所述验证者从针对一个消息c的多个验证模式中选择的一个验证模式的信息；

向所述验证者发送所述多个验证模式的响应信息中的与在接收步骤中接收到的验证模式的信息相对应的响应信息。

26.根据权利要求25所述的认证方法，其中，所述响应信息是使得在如下情况中能够计算出所述私钥s的信息：通过利用所述多个验证模式的响应信息执行的针对消息c的所述多个验证模式的全部已成功。

27.根据权利要求25所述的认证方法，其中，所述响应信息是使用基于所述私钥s和向量$r\in K^n$算出的$z\in K^n$、基于所述向量r和$t\in K^n$算出的$t'\in K^n$和基于将所述向量r代入所述多次多项式f_i的$f_i(r)$和$e_i\in K$算出的$e_i'\in K$而计算出的。

28.根据权利要求25所述的认证方法，其中，所述多次多项式f_i是二次多项式。

29.一种认证方法，包括以下步骤：

从证明者接收消息c；

从针对一个消息c的多个验证模式中选择一个验证模式；

将在选择步骤中所选择的验证模式的信息发送给所述证明者；

从所述证明者接收所述多个验证模式的响应信息中的与在发送步骤中发送的验证模式的信息相对应的响应信息；以及

利用在接收消息c的步骤中所接收的所述消息c以及在接收响应信息的步骤中所接收的所述响应信息来验证所述证明者的合法性，

其中，$s\in K^n$被设置为私钥，并且环K上的多次多项式$f_i(x_1,\cdots,x_n)$被设置为公钥或系统参数，并且$y_i=f_i(s)$被设置为所述公钥，i＝1至m，其中，K^n表示K的n次直积。

30.根据权利要求29所述的认证方法，其中，所述响应信息是使得在如下情况中能够计算出所述私钥s的信息：通过利用所述多个验证模式的响应信息执行的针对消息c的所述多个验证模式的全部已成功。

31.根据权利要求29所述的认证方法，其中，所述响应信息是使用基于所述私钥s和向量$r\in K^n$算出的$z\in K^n$、基于所述向量r和$t\in K^n$算出的$t'\in K^n$和基于将所述向量r代入所述多次多项式f_i的$f_i(r)$和$e_i\in K$算出的$e_i'\in K$而计算出的。

32.根据权利要求29所述的认证方法，其中，所述多次多项式f_i是二次多项式。

33.一种认证方法，包括以下步骤：

将$s\in K^n$设置为私钥，并且将环K上的多次多项式$f_i(x_1,\cdots,x_n)$设置为公钥或系统参数，并且将$y_i=f_i(s)$设置为所述公钥，i＝1至m，其中，K^n表示K的n次直积；

将消息c发送给验证者；

从所述验证者接收答复α；

利用在接收答复α的步骤中所接收的所述答复α来生成要用于验证所述消息c的多项式f”；

将在生成步骤中生成的所述多项式f”发送给所述验证者；

接收由所述验证者从针对一个消息c的k个验证模式中选择的一个验证模式的信息，$k\geq 2$；

向所述验证者发送k种响应信息中的与在接收验证模式的信息的步骤中接收的验证模

式的信息相对应的响应信息。

34.根据权利要求33所述的认证方法，其中，所述响应信息是使得在如下情况中能够计算出所述私钥s的信息：通过利用两种答复α、所述多项式f”以及k种响应信息执行的针对所述消息c的答复与验证模式的总计2k种组合的全部或已成功。

35.根据权利要求33所述的认证方法，其中，所述响应信息是使用基于所述私钥s和向量$r\in K^n$算出的$z\in K^n$、基于所述向量r和$t\in K^n$算出的$t'\in K^n$和基于将所述向量r代入所述多次多项式f_i的$f_i(r)$和$e_i\in K$算出的$e_i'\in K$而计算出的。

36.根据权利要求33所述的认证方法，其中，所述多次多项式f_i是二次多项式。

37.一种认证方法，包括以下步骤：

从证明者接收消息c；

向所述证明者发送答复α；

接收多项式f”，所述多项式f”是由所述证明者利用在发送答复α的步骤中所发送的所述答复α生成的并被用于验证所述消息c；

从针对一个消息c的k个验证模式中选择一个验证模式，$k\geq 2$；

将在选择验证模式的步骤中选择的验证模式的信息发送给所述证明者；

从所述证明者接收k种响应信息中的与在发送验证模式的信息的步骤中所发送的验证模式的信息相对应的响应信息；以及

利用在接收消息c的步骤中所接收的所述消息c、在接收多项式f”的步骤中所接收的所述多项式f”以及在接收响应信息的步骤中所接收的所述响应信息来验证所述证明者的合法性，

其中，$s\in K^n$被设置为私钥，并且环K上的多次多项式$f_i(x_1,\cdots,x_n)$被设置为公钥或系统参数，并且$y_i=f_i(s)$被设置为所述公钥，i＝1至m，其中，K^n表示K的n次直积。

38.根据权利要求37所述的认证方法，其中，所述响应信息是使得在如下情况中能够计算出所述私钥s的信息：通过利用两种答复α、所述多项式f”以及k种响应信息执行的针对所述消息c的答复与验证模式的总计2k种组合的全部已成功。

39.根据权利要求37所述的认证方法，其中，所述响应信息是使用基于所述私钥s和向量$r\in K^n$算出的$z\in K^n$、基于所述向量r和$t\in K^n$算出的$t'\in K^n$和基于将所述向量r代入所述多次多项式f_i的$f_i(r)$和$e_i\in K$算出的$e_i'\in K$而计算出的。

40.根据权利要求37所述的认证方法，其中，所述多次多项式f_i是二次多项式。

41.一种签名生成设备，包括：

密钥设置单元，用于将$s\in K^n$设置为私钥并且将环K上的多次多项式$f_i(x_1,\cdots,x_n)$设置为公钥或系统参数，并将$y_i=f_i(s)$设置为所述公钥，i＝1至m，其中，K^n表示K的n次直积；

消息生成单元，用于基于所述多次多项式$f_i(x_1,\cdots,x_n)$以及所述私钥s来生成N个消息c；

验证模式选择单元，用于基于通过将文档M和所述消息c应用于单向函数而获得的信息，来从k^N个验证模式中选择验证模式，$k\geq 3$；

签名生成单元，用于根据由所述验证模式选择单元所选择的验证模式，来生成数字签名σ，所述数字签名σ将通过使用所述消息c和所述文档M的验证。

42.根据权利要求41所示的签名生成设备，其中，所述数字签名σ是使得在如下情况中

能够计算出所述私钥s的信息：利用与$(k-1)^N+1$个验证模式相对应的数字签名σ执行的验证都已成功。

43.根据权利要求41所述的签名生成设备，其中，所述多次多项式f_i是二次多项式。

44.一种签名验证设备，包括：

文档接收单元，用于接收文档M；

数字签名接收单元，用于接收数字签名σ；

消息获取单元，用于从所述数字签名σ获取N个消息c；以及

数字签名验证单元，用于使用所述消息c和所述文档M，根据基于通过将所述文档M和所述消息c应用于单向函数而获得的信息从k^N个验证模式中所选择的验证模式，对所述数字签名σ进行验证，$k \geq 3$，

其中，$s \in K^n$被设置为私钥，并且环K上的多次多项式$f_i(x_1,\cdots,x_n)$被设置为公钥或系统参数，并且$y_i = f_i(s)$被设置为所述公钥，i＝1至m，其中，K^n表示K的n次直积。

45.根据权利要求44所述的签名验证设备，其中，所述数字签名σ是使得在如下情况中能够计算出所述私钥s的信息：利用与$(k-1)^N+1$个验证模式相对应的数字签名σ执行的验证都已成功。

46.根据权利要求44所述的签名验证设备，其中，所述多次多项式f_i是二次多项式。

认证设备、认证方法和签名生成设备

技术领域

[0001] 本发明涉及认证设备、认证方法、程序和签名生成设备。

背景技术

[0002] 随着信息处理技术和通信技术的快速发展，公务和私人文档的数字化正在快速推进。因此，许多个人和公司对电子文档的安全管理非常感兴趣。随着兴趣的增加，在许多场合已开始热烈地讨论针对电子文档的诸如窃听和伪造之类的篡改的安全性。例如通过对电子文档加密来确保针对电子文档的窃听的安全性。此外，例如通过利用数字签名来确保针对电子文档的伪造的安全性。然而，加密和数字签名必须是能充分地防篡改的。

[0003] 数字签名用来指定电子文档的作者。因此，数字签名应当能够仅由该电子文档的作者来生成。如果恶意第三方能够生成相同的数字签名，则这样的第三方可以冒充电子文档的作者。即，电子文档会被恶意第三方伪造。对于防止这样的伪造的数字签名安全性，已经提出了各种意见。作为当前广泛使用的数字签名方案，例如已知了使用RSA签名方案和DSA签名方案的方案。

[0004] RSA签名方案取用“对大的合成数进行质因数分解的难度(以下称为质因数分解问题)”作为安全性的基础。此外，DSA签名方案取用“求解离散对数问题的难度”作为安全性的基础。这些基础是基于通过利用典型计算机来高效地求解质因数问题和离散对数问题的算法并不存在。即，上面提到的难度意味着典型计算机的计算难度。然而，据说，当使用量子计算机时，可以高效地计算出质因数分解问题和离散对数问题的解。

[0005] 与RSA签名方案和DSA签名方案类似地，当前使用的许多数字签名方案和公钥认证方案也取用了质因数分解问题或离散对数问题的难度作为安全性的基础。因此，如果量子计算机投入实用，则将不再确保这些数字签名方案和公钥认证方案的安全性。因此，希望具有新的数字签名方案和公钥认证方案，其采用与诸如可被量子计算机容易求解出的质因数分解问题和离散对数问题之类的问题不同的问题作为安全性的基础。作为不容易被量子计算机求解出的问题，例如，在求解多元多项式上存在难度(以下称为多元多项式问题)。

[0006] 例如，作为采用多元多项式问题作为安全性基础的数字签名方案，已知了基于Matsumoto-Imai(MI)密码术、Hidden Field Equation(HFE)密码术、Oil-Vinegar(OV)签名方案以及Tamed Transformation法(TTM)密码术的这些方案。例如，基于HFE的数字签名方案在Jacques Patarin的Asymmetric Cryptography with a Hidden Monomial，CRYPTO 1996，pp.45-60以及Patarin，J.，Courtois，N.和Goubin，L.的QUARTZ，128-Bit Long Digital Signatures，In Naccache，D.，Ed.Topics in Cryptology-CT-RSA 2001(San Francisco，CA，USA，April 2001)，vol.2020 of Lecture Notes in Computer Science，Springer-Verlag，pp.282-297中被公开。

发明内容

[0007] 如上，多元多项式问题是称为NP难题的、即使利用量子计算机也难以求解出的问

题的示例。通常，利用由HFE代表的多元多项式问题的公钥认证方案使用具有特殊陷门(trapdoor)的多次多元联立方程式。例如，与$x_1,\cdots,x_n$有关的多次多元联立方程式$F(x_1,\cdots,x_n)=y$以及线性变换A和B被提供，并且线性变换A和B秘密地被管理。在此情况中，多次多元联立方程式F以及线性变换A和B是陷门。

[0008] 知道陷门F、A和B的实体可以求解与$x_1,\cdots,x_n$有关的方程式$B(F(A(x_1,\cdots,x_n)))=y'$。另一方面，与$x_1,\cdots,x_n$有关的方程式$B(F(A(x_1,\cdots,x_n)))=y'$不能由不知道陷门F、A和B的实体求解出。通过利用该机制，可以实现采用多次多元联立方程式作为安全性基础的公钥认证方案和数字签名方案。

[0009] 如上，为了实现这样的公钥认证方案和数字签名方案，需要使用满足$B(F(A(x_1,\cdots,x_n)))=y$的特殊多次多元联立方程式。然而，根据现有方案，需要在签名生成时求解多次多元联立方程式F，因此仅那些可被容易求解出的方程式可被用作F。即，根据这些现有方案，仅可以使用通过组合可以较容易地求解出的三个函数(陷门)B、F和A而形成的多次多元联立方程式$B(F(A(x_1,\cdots,x_n)))=y$，因此难以确保足够的安全性。结果，对于使用存在高效求解手段(陷门)的多次多元联立方程式的公钥认证方案和数字签名方案，提出了大量的攻击方法，因此希望设法使得没有高效求解手段(陷门)的多次多元联立方程式被用于公钥认证方案和数字签名方案。

[0010] 鉴于前面的状况，希望提供能够实现通过利用没有高效求解手段(陷门)的多次多元联立方程式而增强其安全性的公钥认证方案或数字签名方案的、新颖的经改进的认证设备、认证方法、程序和签名生成设备。

[0011] 根据本发明的一个实施例，提供了一种认证设备，该设备包括：密钥设置单元，用于将$s\in K^n$设置为私钥并且将环K上的多次多项式$f_i(x_1,\cdots,x_n)$以及$y_i=f_i(s)$设置为公钥，$i=1$至m；消息发送单元，用于将消息c发送给验证者；验证模式接收单元，用于接收关于验证者从针对一个消息c的k个验证模式中选择的一个验证模式的信息，$k\geq 3$；响应发送单元，用于向验证者发送k种响应信息中的与由验证模式接收单元接收到的验证模式有关的信息所对应的响应信息。响应信息是这样的信息，该信息使得在通过利用k种响应信息执行的针对消息c的所有k个验证模式都成功的情况中能够计算出私钥s。

[0012] 该认证设备可被配置为使得：在执行以下步骤时：由消息发送单元发送一个或多个消息c的第一步、由验证模式接收单元针对每个消息c从验证者接收与验证模式有关的信息的第二步、以及由响应发送单元针对与验证模式有关的每个信息发送响应信息的第三步，如果验证者针对所有响应信息成功地执行了验证，则认证成功。

[0013] 该认证设备可被配置为使得：执行以下步骤的处理被重复：由消息发送单元发送一个或多个消息c的第一步、由验证模式接收单元针对每个消息c从验证者接收与验证模式有关的信息的第二步、以及由响应发送单元针对与验证模式有关的每个信息来发送响应信息的第三步，并且当执行了第一步至第三步预定次数时，如果验证者针对所有响应信息每次都成功地 执行了验证，则认证成功。

[0014] 在消息$c=(c_1,\cdots,c_m)$的情况中，消息发送单元可以通过利用单向函数H来计算新的消息$c'=H(c)$，并且将消息c'发送给验证者，并且响应发送单元可以将验证者即使利用响应信息也不能恢复出来的消息c的元素与响应信息一起发送。

[0015] 根据本发明另一实施例，提供了一种认证设备，包括：消息接收单元，用于从证明

者接收消息c；验证模式选择单元，用于从针对一个消息c的k个验证模式中选择一个验证模式，$k \geq 3$；验证模式发送单元，用于将与验证模式选择单元所选择的验证模式有关的信息发送给证明者；响应接收单元，用于从证明者接收k种响应信息中的与由验证模式发送单元所发送的验证模式有关的信息所对应的响应信息；以及验证单元，用于利用由消息接收单元接收的消息c以及由响应接收单元接收的响应信息来验证证明者的合法性。$s \in K^n$被设置为私钥并且环K上的多次多项式$f_i(x_1,\cdots,x_n)$以及$y_i = f_i(s)$被设置为公钥，i＝1至m。响应信息是这样的信息，该信息使得在通过利用k种响应信息执行的针对消息c的所有k个验证模式都成功的情况中能够计算出私钥s。

[0016] 根据本发明另一实施例，提供了一种认证设备，包括：密钥设置单元，用于将$s \in K^n$设置为私钥并且将环K上的多次多项式$f_i(x_1,\cdots,x_n)$以及$y_i = f_i(s)$设置为公钥，i＝1至m；消息发送单元，用于将消息c发送给验证者；答复接收单元，用于从验证者接收答复α；多项式生成单元，用于利用答复接收单元所接收的答复α来生成将用于验证消息c的多项式f”；多项式发送单元，用于将由多项式生成单元生成的多项式f”发送给验证者；验证模式接收单元，用于接收与验证者从针对一个消息c的k个验证模式中选择的一个验证模式有关的信息，$k \geq 2$；响应发送单元，用于向验证者发送k种响应信息中的与由验证模式接收单元接收到的验证模式有关的信息所对应的响应信息。响应信息是这样的信息，该信息使得在通过利用两种答复α、多项式f”以及k种响应信息执行的针对消息c的答复与验证模式的所有总计2k种组合都成功的情况中，能够计算出私钥s。

[0017] 该认证设备可被配置为使得：在执行以下步骤时：由消息发送单元发 送一个或多个消息c的第一步、由答复接收单元接收针对每个消息c的答复α的第二步、由多项式生成单元利用在第二步中接收的答复α来生成多项式并且由多项式发送单元发送多项式f”的第三步，由验证模式接收单元针对每个消息c从验证者接收与验证模式有关的信息的第四步、以及由响应发送单元针对与验证模式有关的每个信息来发送响应信息的第五步，如果验证者针对所有响应信息成功地执行了验证，则认证成功。

[0018] 该认证设备可被配置为使得：执行以下步骤的处理被重复：由消息发送单元发送一个或多个消息c的第一步、由答复接收单元接收针对每个消息c的答复α的第二步、由多项式生成单元利用在第二步中接收的答复α来生成多项式并且由多项式发送单元发送多项式f”的第三步，由验证模式接收单元针对每个消息c从验证者接收与验证模式有关的信息的第四步、以及由响应发送单元针对与验证模式有关的每个信息来发送响应信息的第五步，并且当执行了第一步至第五步预定次数时，如果验证者针对所有响应信息每次都成功地执行了验证，则认证成功。

[0019] 根据本发明另一实施例，提供了一种认证设备，包括：消息接收单元，用于从证明者接收消息c；答复发送单元，用于向证明者发送答复α；多项式接收单元，用于接收多项式f”，多项式f”是由证明者利用由答复发送单元发送的答复α生成的并被用于验证消息c；验证模式选择单元，用于从针对一个消息c的k个验证模式中选择一个验证模式，$k \geq 2$；验证模式发送单元，用于将与验证模式选择单元所选择的验证模式有关的信息发送给证明者；响应接收单元，用于从证明者接收k种响应信息中的与由验证模式发送单元所发送的验证模式有关的信息所对应的响应信息；以及验证单元，用于利用由消息接收单元接收的消息c、由多项式接收单元接收的多项式f”以及由响应接收单元接收的响应信息来验证证明者的

合法性。$s \in K^n$被设置为私钥并且环K上的多次多项式$f_i(x_1,\cdots,x_n)$以及$y_i = f_i(s)$被设置为公钥，i＝1至m。响应信息是这样的信息，该信息使得在通过利用两种答复α、多项式f”以及k种响应信息执行的针对消息c的答复与验证模式的所有总计2k种组合都成功的情况中，能够计算出私钥s。

[0020] 根据本发明另一实施例，提供了一种认证方法，包括以下步骤：将 $s \in K^n$设置为私钥并且将环K上的多次多项式$f_i(x_1,\cdots,x_n)$以及$y_i = f_i(s)$设置为公钥，i＝1至m；将消息c发送给验证者；接收与验证者从针对一个消息c的k个验证模式中选择的一个验证模式有关的信息，$k \geq 3$；向验证者发送k种响应信息中的与在接收步骤中接收到的验证模式有关的信息所对应的响应信息。响应信息是这样的信息，该信息使得在通过利用k种响应信息执行的针对消息c的所有k个验证模式都成功的情况中能够计算出私钥s。

[0021] 根据本发明另一实施例，提供了一种认证方法，包括以下步骤：从证明者接收消息c；从针对一个消息c的k个验证模式中选择一个验证模式，$k \geq 3$；将与在选择步骤中所选择的验证模式有关的信息发送给证明者；从证明者接收k种响应信息中的与在发送步骤中所发送的验证模式有关的信息所对应的响应信息；以及利用在接收消息c的步骤中所接收的消息c以及在接收响应信息的步骤中所接收的响应信息来验证证明者的合法性。$s \in K^n$被设置为私钥并且环K上的多次多项式$f_i(x_1,\cdots,x_n)$以及$y_i = f_i(s)$被设置为公钥，i＝1至m。响应信息是这样的信息，该信息使得在通过利用k种响应信息执行的针对消息c的所有k个验证模式都成功的情况中能够计算出私钥s。

[0022] 根据本发明另一实施例，提供了一种认证方法，包括以下步骤：将$s \in K^n$设置为私钥并且将环K上的多次多项式$f_i(x_1,\cdots,x_n)$以及$y_i = f_i(s)$设置为公钥，i＝1至m；将消息c发送给验证者；从验证者接收答复α；利用在接收答复α的步骤中所接收的答复α来生成将用于验证消息c的多项式f”；将在生成步骤中生成的多项式f”发送给验证者；接收与验证者从针对一个消息c的k个验证模式中选择的一个验证模式有关的信息，$k \geq 2$；向验证者发送k种响应信息中的与在接收与验证模式有关的信息的步骤中接收到的与验证模式有关的信息相对应的响应信息。响应信息是这样的信息，该信息使得在通过利用两种答复α、多项式f”以及k种响应信息执行的针对消息c的答复与验证模式的所有总计2k种组合都成功的情况中，能够计算出私钥s。

[0023] 根据本发明另一实施例，提供了一种认证方法，包括以下步骤：从证 明者接收消息c；向证明者发送答复α；接收多项式f”，多项式f”是由证明者利用在发送答复α的步骤中所发送的答复α生成的并被用于验证消息c；从针对一个消息c的k个验证模式中选择一个验证模式，$k \geq 2$；将与在选择验证模式的步骤中所选择的验证模式有关的信息发送给证明者；从证明者接收k种响应信息中的与在发送与验证模式有关的信息的步骤中所发送的与验证模式有关的信息相对应的响应信息；以及利用在接收消息c的步骤中所接收的消息c、在接收多项式f”的步骤中所接收的多项式f”以及在接收响应信息的步骤中所接收的响应信息来验证证明者的合法性。$s \in K^n$被设置为私钥并且环K上的多次多项式$f_i(x_1,\cdots,x_n)$以及$y_i = f_i(s)$被设置为公钥，i＝1至m。响应信息是这样的信息，该信息使得在通过利用两种答复α、多项式f”以及k种响应信息执行的针对消息c的答复与验证模式的所有总计2k种组合都成功的情况中，能够计算出私钥s。

[0024] 根据本发明另一实施例，提供了一种使得计算机实现如下功能的程序：密钥设置

功能，用于将$s \in K^n$设置为私钥并且将环K上的多次多项式$f_i(x_1,\cdots,x_n)$以及$y_i = f_i(s)$设置为公钥，i＝1至m；消息发送功能，用于将消息c发送给验证者；验证模式接收功能，用于接收与验证者从针对一个消息c的k个验证模式中选择的一个验证模式有关的信息，$k \geq 3$；响应发送功能，用于向验证者发送k种响应信息中的与由验证模式接收功能接收到的验证模式有关的信息所对应的响应信息。响应信息是这样的信息，该信息使得在通过利用k种响应信息执行的针对消息c的所有k个验证模式都成功的情况中能够计算出私钥s。

[0025] 根据本发明另一实施例，提供了一种记录有该程序的计算机可读记录介质。

[0026] 根据本发明另一实施例，提供了一种签名生成设备，包括：密钥设置单元，用于将$s \in K^n$设置为私钥并且将环K上的多次多项式$f_i(x_1,\cdots,x_n)$以及$y_i = f_i(s)$设置为公钥，i＝1至m；消息生成单元，用于基于多次多项式$f_i(x_1,\cdots,x_n)$以及私钥s来生成N个消息c；验证模式选择单元，用于基于通过将文档M和消息c应用于单向函数H而获得的信息，来从k^N个验证模式中选择验证模式，$k \geq 3$；签名生成单元，用于根据由验证模式选择单 元所选择的验证模式，来生成将通过使用消息c和文档M的验证的数字签名σ。数字签名σ是这样的信息，该信息使得在利用与$(k-1)^N+1$个验证模式相对应的数字签名σ执行的验证都成功的情况下，能够计算出私钥s。

[0027] 对于m和n，可以m＜n。对于m和n还可以$2^{m-n} << 1$。

[0028] 根据上述本发明的实施例，可以实现通过利用不存在高效求解手段(陷门)的多次多元联立方程式增强了其安全性的公钥认证方案或数字签名方案。

附图说明

[0029] 图1是用于描述一般的公钥认证方案的算法结构的说明图；

[0030] 图2是用于描述数字签名方案的算法结构的说明图；

[0031] 图3是用于描述n次传递公钥认证方案的说明图；

[0032] 图4是用于描述本发明第一实施例(3次传递)的公钥认证方案的算法的说明图；

[0033] 图5是用于描述根据本实施例的公钥认证方案的扩展算法的说明图；

[0034] 图6是用于描述根据本实施例的公钥认证方案的并行算法的说明图；

[0035] 图7是用于描述根据本实施例的公钥认证方案的非交互式算法的说明图；

[0036] 图8是用于描述根据本实施例的公钥认证方案的具体算法的说明图；

[0037] 图9是用于描述根据本实施例的公钥认证方案的高效算法的说明图；

[0038] 图10是用于描述根据本实施例的公钥认证方案的高效算法(修改A)的说明图；

[0039] 图11是用于描述根据本实施例的公钥认证方案的高效算法(修改B)的说明图；

[0040] 图12是用于描述本发明第二实施例(5次传递)的公钥认证方案的算法的说明图；

[0041] 图13是用于描述根据本实施例的公钥认证方案的扩展算法的说明图；

[0042] 图14是用于描述根据本实施例的公钥认证方案的并行算法的说明 图；

[0043] 图15是用于描述根据本实施例的公钥认证方案的扩展算法的并行化的说明图；

[0044] 图16是用于描述根据本实施例的公钥认证方案的非交互式算法的说明图；

[0045] 图17是用于描述根据本实施例的公钥认证方案的具体算法的说明图；

[0046] 图18是用于描述根据本实施例的公钥认证方案的高效算法的说明图；

[0047] 图19是用于描述根据本实施例的公钥认证方案的高效算法(修改A)的说明图；

[0048] 图20是用于描述根据本实施例的公钥认证方案的高效算法(修改B)的说明图；

[0049] 图21是用于描述根据本实施例的公钥认证方案的高效算法(修改C)的说明图；

[0050] 图22是用于描述根据本实施例的公钥认证方案的高效算法(修改D)的说明图；

[0051] 图23是用于描述根据本实施例的公钥认证方案的高效算法(修改E)的说明图；

[0052] 图24是用于描述根据本实施例的公钥认证方案的高效算法(修改F)的说明图；

[0053] 图25是用于描述根据本实施例的公钥认证方案的高效算法(修改G)的说明图；

[0054] 图26是用于描述能够执行根据本发明每个实施例的算法的信息处理装置的示例硬件配置的说明图；

[0055] 图27是比较根据本发明第一和第二实施例的公钥认证方案的效率的图表；以及

[0056] 图28是用于描述设置参数的合适方法、根据本发明第一和第二实施例的公钥认证方案所使用的方法，及其效果。

具体实施方式

[0057] 下面，将参考附图详细描述本发明的优选实施例。注意，在本说明书和附图中，用相同的标号来表示具有基本上相同的功能和结构的结构元件，并且省略对这些结构元件的重复描述。

[0058] [描述的流程]

[0059] 这里将简要提及与后面描述的本发明的实施例有关的描述的流程。首先，将参考图1描述公钥认证方案的算法结构。接下来，将参考图2描述数字签名方案的算法结构。然后，将参考图3描述n次传递(n-pass)公钥认证方案。另外，在描述本发明的实施例之前，还要简要描述一般的HFE签名方案。

[0060] 接下来，参考图4描述根据本发明第一实施例(3次传递)的公钥认证方案的算法。然后，参考图5描述用于根据本实施例的公钥认证方案的扩展算法。然后，参考图6描述用于根据本实施例的公钥认证方案的并行算法。接下来，参考图7描述用于根据本实施例的公钥认证方案的非交互式算法。然后，参考图8描述用于根据本实施例的公钥认证方案的具体算法。然后，参考图9至11描述用于根据本实施例的公钥认证方案的高效算法。

[0061] 接下来，参考图12描述根据本发第二实施例(5次传递)的钥认证方案的算法。然后，参考图13描述用于根据本实施例的公钥认证方案的扩展算法。然后，参考图14和15描述用于根据本实施例的公钥认证方案的并行算法。接下来，参考图16描述用于根据本实施例的公钥认证方案的非交互式算法。然后，参考图17描述用于根据本实施例的公钥认证方案的具体算法。然后，参考图18至25描述用于根据本实施例的公钥认证方案的高效算法。

[0062] 接下来，描述将根据本发明第一和第二实施例的高效算法应用于二次或更高次的多元多项式的扩展方法。然后，参考图26描述能够实现本发明第一和第二实施例中的每个算法的信息处理装置的示例硬件配置。最 后，将总结实施例的技术思想并且简要描述这些技术思想获得的效果。

[0063] (描述项目)

[0064] 1：介绍

[0065] 1-1：公钥认证方案的算法结构

[0066] 1-2：数字签名方案的算法结构

[0067] 1-3:N次传递公钥认证方案

[0068] 1-4:HFE数字签名方案

[0069] 1-4-1:HFE函数的性质

[0070] 1-4-2:HFE数字签名方案的算法

[0071] 2:第一实施例

[0072] 2-1:用于公钥认证方案的算法

[0073] 2-2:扩展算法

[0074] 2-3:并行算法

[0075] 2-4:非交互式算法

[0076] 2-5:修改为数字签名方案

[0077] 2-6:具体示例

[0078] 2-7:高效算法

[0079] 2-8:多次多元联立方程式的形式

[0080] 2-8-1:与共同密钥块密码有关的形式

[0081] 2-8-2:与哈希函数有关的形式

[0082] 2-8-3:与流密码有关的形式

[0083] 3:第二实施例

[0084] 3-1:用于公钥认证方案的算法

[0085] 3-2:扩展算法

[0086] 3-3:并行算法

[0087] 3-4:非交互式算法

[0088] 3-5:修改为数字签名方案

[0089] 3-6:具体示例

[0090] 3-7:高效算法

[0091] 4:高效算法的一般化

[0092] 5:示例硬件配置

[0093] 6:总结

[0094] <1:介绍>

[0095] 首先,在详细描述本发明的实施例之前,将简要描述一般的公钥认证方案的算法结构、一般的数字签名方案的算法结构以及n次传递公钥认证方案。

[0096] [1-1:公钥认证方案的算法结构]

[0097] 首先,参考图1描述一般的公钥认证方案的算法结构。图1是用于描述一般的公钥认证方案的算法结构的说明图。

[0098] (概述)

[0099] 公钥认证方案是这样的认证方案,其中,一人(证明者)利用公钥pk和私钥sk来使另一人(验证者)相信其是证明者本身。例如,使得证明者A的公钥pk_A为验证者所知。另一方面,证明者A的私钥sk_A由证明者秘密地管理。根据公钥认证方案,知道与公钥pk_A相对应的私钥sk_A的人被认为是证明者A本身。

[0100] 在证明者A试图向验证者B证明其是证明者本身的情况中,证明者A可以与验证者B

执行交互式协议，并且证明其知道与公钥pk_A相对应的私钥sk_A。然后，在验证者B通过该交互式协议证实了证明者A知道私钥sk_A的情况中，则证明者A的合法性（其是证明者本身）得到证实。

[0101] 另外，为了确保公钥认证方案的安全性，要满足下面阐述的两个条件。

[0102] 第一条件是尽可能降低在交互式协议被执行时没有私钥sk_A的伪造者使伪造成立的概率。满足该第一条件被称为“合理性”（soundness）。换言之，通过合理的交互式协议，没有私钥sk_A的伪造者不会以不可忽略的概率来使伪造成立。第二条件是即使交互式协议被执行，与证明者A的私 钥sk_A有关的信息一点儿也不会被泄露给验证者B。满足该第二条件被称为“零知识”。

[0103] 通过利用具有如上所述的合理性和零知识的交互式协议来确保公钥认证方案的安全性。

[0104] （模型）

[0105] 在公钥认证方案的模型中，存在如图1所示的两个实体，即证明者和验证者。证明者通过利用密钥生成算法Gen来生成该证明者独有的公钥pk和私钥sk对。然后，证明者通过利用使用密钥生成算法Gen生成的该公钥pk和私钥sk对来与验证者执行交互式协议。此时，证明者通过利用证明者算法P来执行交互式协议。如上所述，在交互式协议中，证明者通过利用证明者算法P来向验证者证明其拥有私钥sk。

[0106] 另一方面，验证者通过利用验证者算法V来执行交互式协议，并且验证该证明者是否拥有与该证明者已公开的公钥相对应的私钥。即，验证者是验证证明者是否用于与公钥相对应的私钥的实体。如上所述，公钥认证方案的模型由两个实体构成，即，由证明者和验证者，以及三个算法，即密钥生成算法Gen、证明者算法P和验证者算法V构成。

[0107] 另外，表述“证明者”和“验证者”将被用在下面的描述中，但是这些表述严格来说是指实体。因此，执行密钥生成算法Gen和证明者算法P的主体是与实体“证明者”相对应的信息处理装置。类似地，执行验证者算法V的主体是信息处理装置。这些信息处理装置的硬件配置例如如图26所示。即，密钥生成算法Gen、证明者算法P和验证者算法V是由CPU 902基于记录在ROM 904、RAM 906、存储单元920、可移除记录介质928等上的程序来执行的。

[0108] （密钥生成算法Gen）

[0109] 密钥生成算法Gen由证明者使用。密钥生成算法Gen是用于生成证明者独有的公钥pk和私钥sk对的算法。通过密钥生成算法Gen生成的公钥pk被公开。此外，所公开的公钥pk由验证者使用。另一方面，通过密钥生成算法Gen生成的私钥sk由证明者秘密地管理。被秘密地管理的私钥 sk用来向验证者证明对与公钥pk相对应的私钥sk的拥有。形式上，密钥生成算法Gen被表示为下面的公式（1），作为获取安全性参数1^{λ}（λ是等于或大于0的整数）作为输入并输出私钥sk和公钥pk的算法。

[0110] [表达式1]

[0111] $$(sk,pk)\leftarrow Gen(1^{\lambda}) \qquad \cdots(1)$$

[0112] （证明者算法P）

[0113] 证明者算法P由证明者使用。证明者算法P是用于证明对与公钥pk相对应的私钥sk的拥有的算法。证明者算法P被定义为获取证明者的公钥pk和私钥sk作为输入并且执行与验证者的交互式协议的算法。

[0114] (验证者算法V)

[0115] 验证者算法V由验证者使用。验证者算法V是用于在交互式协议中验证证明者是否拥有与公钥pk相对应的私钥sk的算法。验证者算法V被定义为获取证明者的公钥pk作为输入并在执行了与证明者的交互式协议之后输出0或1(1比特)的算法。此外,在输出0的情况中,假设证明者是非法的,并且在输出1的情况中,假设证明者是合法的。形式上,验证者算法V被表示为如下的公式(2)。

[0116] [表达式2]

[0117] $0/1 \leftarrow V(pk,m,\sigma)$ …(2)

[0118] (补充)

[0119] 如上所述,公钥认证方案必须满足两个条件,即,合理性和零知识,以确保安全性。然而,为了使得证明者能证明其拥有私钥sk,需要证明者执行依赖于私钥sk的过程,向验证者通知结果并且使验证者基于所通知内容执行验证。执行依赖于私钥sk的过程是必要的,以保证合理性。另一方面,即使在过程的结果被通知给验证者时,也需要与私钥sk_A有关的信息一点儿也不会被泄露给验证者。因此,需要将密钥生成算法Gen、证明者算法P和验证者算法V设计为满足这些项。

[0120] 在前面,已描述了一般的公钥认证方案的算法结构。

[0121] [1-2:数字签名方案的算法结构]

[0122] 接下来,将参考图2描述数字签名方案的算法结构。图2是用于描述一般的数字签名方案的算法结构的说明图。

[0123] (概述)

[0124] 与纸质文档不同,无法对数字化数据进行密封或签名。因此,为了证明数字化数据的作者的合法性,产生与对纸质文档进行密封或签名相同效果的电子机制成为必要。该机制就是数字签名。例如,将仅数据的作者知道的签名数据提供给与该数据相关联的接收者并且在接收者侧验证该签名数据的机制被称为数字签名方案。

[0125] (模型)

[0126] 在数字签名方案的模型中,存在如图2所示的两个实体,即签名者和验证者。此外,数字签名方案的模型还由三个算法,即密钥生成算法Gen、签名生成算法Sig和签名验证算法Ver构成。

[0127] 签名者通过利用密钥生成算法Gen来生成该签名者独有的验证密钥pk和签名密钥sk对。此外,签名者通过利用签名生成算法Sig来生成将被添加到文档M的数字签名σ。即,签名者是向文档M添加数字签名的实体。另一方面,验证者通过利用签名验证算法Ver来验证添加到文档M的数字签名σ。即,验证者是验证该数字签名σ以检查文档M的作者是否是该签名者的实体。

[0128] 另外,表述“签名者”和“验证者”将被用在下面的描述中,但是这些表述严格来说是指实体。因此,执行密钥生成算法Gen和签名生成算法Sig的主体是与实体“签名者”相对应的信息处理装置。类似地,执行签名验证算法Ver的主体也是信息处理装置。这些信息处理装置的硬件配置例如如图26所示。即,密钥生成算法Gen、签名生成算法Sig和签名验证算法Ver是由CPU 902基于记录在ROM 904、RAM 906、存储单元920、可移除记录介质928等上的程序来执行的。

[0129] (密钥生成算法Gen)

[0130] 密钥生成算法Gen由签名者使用。密钥生成算法Gen是用于生成签名 者独有的验证密钥pk和签名密钥sk对的算法。通过密钥生成算法Gen生成的验证密钥pk被公开。另一方面，通过密钥生成算法Gen生成的签名密钥sk由签名者秘密地管理。由签名者秘密地管理的签名密钥sk被用来生成添加到文档M的数字签名σ。形式上，该密钥生成算法Gen被表示为如下的公式(3)，作为获取安全性参数1^{λ}(λ是等于或大于0的整数)作为输入并输出私钥sk和公钥pk的算法。

[0131] [表达式3]

[0132] $(sk,pk)\leftarrow Gen(1^{\lambda})$ …(3)

[0133] (签名生成算法Sig)

[0134] 签名生成算法Sig由签名者使用。签名生成算法Sig是用于生成将被添加到文档M的数字签名σ的算法。形式上，签名生成算法Sig被标识为如下的公式(4)，作为获取文档M和签名者的签名密钥sk作为输入并输出数字签名σ的算法。

[0135] [表达式4]

[0136] $\sigma\leftarrow Sig(sk,M)$ …(4)

[0137] (签名验证算法Ver)

[0138] 签名验证算法Ver由验证者使用。签名验证算法Ver是用于验证数字鉴名σ是否是文档M的有效数字签名的算法。形式上，签名验证算法Ver被表示为如下的公式(5)，作为获取验证密钥pk、文档M和签名者的数字签名σ作为输入并输出0或1(1比特)的算法。此外，在0被输出的情况中(在公钥pk否决文档M和数字签名σ的情况中)，文档M的数字签名σ是无效的。在1被输出的情况中(在公钥pk接受文档M和数字签名σ的情况中)，文档M的数字签名σ是有效的。

[0139] [表达式5]

[0140] $0/1\leftarrow Ver(pk,M,\sigma)$ …(5)

[0141] 在前面，已描述了一般的数字签名方案的算法结构。

[0142] [1-3:N次传递公钥认证方案]

[0143] 接下来，将参考图3描述n次传递公钥认证方案。图3是用于描述n 次传递公钥认证方案的说明图。

[0144] 如上所述，公钥认证方案是用于在交互式协议中向验证者证明证明者拥有与公钥pk相对应的私钥sk的认证方案。此外，为了保证公钥认证方案的安全性，必须满足两个条件，即合理性和零知识。因此，如图3所示，在交互式协议中，在证明者和验证者两者正执行这些处理的执行时，在证明者和验证者之间执行信息交换n次。

[0145] 在n次传递公钥认证方案的情况中，由证明者利用证明者算法P来执行处理(步骤1)，并且信息T_1被发送给验证者。接下来，由验证者利用验证者算法V执行处理(步骤2)，并且信息T_2被发送给证明者。处理(步骤3，…，步骤n)以类似方式被执行并且多个信息T_3，…，T_n被发送，并且处理(步骤n+1)被执行。将基于多个信息被发送/接收n次的交互式协议的这样的公钥认证方案称为“n次传递”公钥认证方案。

[0146] 在前面，已描述了n次传递公钥认证方案。

[0147] [1-4:HFE数字签名方案]

[0148] 这里，将简要描述HFE数字签名方案作为使用多次多元联立方程式的数字签名方案的示例。

[0149] (1-4-1：HFE函数的性质)

[0150] 首先，将简要描述HFE函数F_t的定义和HFE函数F_t的性质。

[0151] 《符号的定义》

[0152] K：由包括q个数的元素形成的有限环

[0153] K^n：K的N次直积

[0154] $F_t: K^n \rightarrow K^n$

[0155] A：有限环K的N次扩展(元素数q^n)

[0156] B：有限环K的M次扩展(元素数q^m)

[0157] ：线性映射$A \rightarrow K^n$(参见下面的公式(6))

[0158] S：对K^n的逆仿射变换

[0159] T：对K^n的逆仿射变换

[0160] f：中心映射(参见下面的公式(7))

[0161] 陷门：S，T，a_{ij}，b_i，c

[0162] [表达式6]

[0163] $\phi(x_0+x_1X+\cdots+x_{n-1}X^{n-1})=(x_0,\cdots,x_{n-1})$ …(6)

[0164] [表达式7]

[0165]

$$f: A \rightarrow A, X \mapsto \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} a_{ij} X^{q^i+q^j} + \sum_{i=0}^{n-1} b_i X^{q^i} + c \quad \ldots(7)$$

[0166] 另外，当d为不太大的整数时，则a_{ij}，b_i，c∈A，“如果$q^i+q^j>d$，则$a_{ij}=0$”并且“如果$q^i>d$，则$b_j=0$”。

[0167] 《HFE函数F_t的结构》

[0168] HFE函数F_t通过如下的合成映射$F_t=T*F*S$来表达，该合成映射是通过变换S的映射、中心映射F $(F=\varphi^{-1}*f*\varphi)$以及通过变换T的映射(*是映射的合成)。用于计算$y=F_t(x)$的算法如下。

[0169] (步骤1)通过变换S将给定的$x=(x_0,\cdots,x_{n-1})\in K^n$变换为$x'=(x_0',\cdots,x_{n-1}')\in K^n$。

[0170] (步骤2)通过 φ^{-1}将$x'\in K^n$变换为$X'\in A$。

[0171] (步骤3)通过中心映射f将$X'\in A$变换为$Y'=f(X')\in A$。

[0172] (步骤4)通过 φ将$Y'\in A$变换为$y'=(y_0',\cdots,y_{n-1}')\in K^n$。

[0173] (步骤5)通过变换T将$y'\in K^n$变换为$y=(y_0,\cdots,y_{n-1})\in K^n$。

[0174] (步骤6)输出$y\in K^n$。

[0175] 如上面的公式(7)所示，HFE函数F_t包括基于一个变量的非线性多项式的中心映射f。因此，存在这样的可能性：存在与上域(codomain)A的元素Y'的单变量多项式的根的集合相对应的前像(preimage)$\{Z\in A \mid f(Z)=Y'\}$的不同数目的元素。在此情况中，与HFE函数F_t有关的前像的元素个数对于值域的元素y来说是多个。

[0176] 还存在这样的可能性：对于上域A的元素Y'，前像{Z∈A|f(Z)＝Y'}的元素不存在。在此情况中，上域的该元素(对于其来说前像{Z∈A|f(Z)＝Y'}的元素不存在)不被包括在值域中，并且因此上域和值域是不同的。

[0177] 在下面，将更详细描述针对HFE函数F_t的计算算法。

[0178] 《前向计算算法》

[0179] 用于HFE函数F_t的前向计算算法由这样的步骤形成：将给定的$x\in K^n$代入HFE函数$F_t(x)$中并获得$y=F_t(x)\in K^n$。当定义域的一个元素x被输入到前向计算算法中时，则值域的一个元素y被输出。

[0180] 《后向计算算法》

[0181] 与HFE函数F_t有关的后向计算算法由下面的步骤1至步骤7形成。

[0182] (步骤1)通过将给定的$y=(y_0,\cdots,y_{n-1})\in K^n$应用于变换T的逆变换$T^{-1}$来获得$y'=(y_0',\cdots,y_{n-1}')\in K^n$。

[0183] (步骤2)将$y'=(y_0',\cdots,y_{n-1}')\in K^n$变换为 φ^{-1}。

[0184] (步骤3)利用Y'来计算$X'\in\{Z\in A|f(Z)=Y'\}$的集合。这里，在$X'\in\{Z\in A|f(Z)=Y'\}$为空集的情况中，例外值err被输出。另外，$X'\in\{Z\in A|f(Z)=Y'\}$例如是通过对多项式f(X)-Y'进行因数分解而获得的。此外，当上域的元素Y'随机地被选择时，针对元素Y'的前像$\{Z\in A|f(Z)=Y'\}$具有m个元素的概率大约为1/(m!e)(这里，e是Napier常数)。

[0185] (步骤4)从$X'\in\{Z\in A|f(Z)=Y'\}$的集合中选择一个元素X'。

[0186] (步骤5)在步骤4中选出的一个元素$X'\in A$通过 φ被变换为$x'=(x_0',\cdots,x_{n-1}')\in K^n$。

[0187] (步骤6)通过变换S的逆变换S^{-1}将$x'\in K^n$变换为$x=(x_0,\cdots,x_{n-1})\in K^n$。

[0188] (步骤7)输出$x\in K^n$。

[0189] 注意，在上面的步骤3中，存在这样的可能性：$X'\in\{Z\in A|f(Z)=Y'\}$的元素数目$\alpha=|\{Z\in A|f(Z)=Y'\}|$为$\alpha=0$或$\alpha\geq 2$。

[0190] (1-4-2：HFE数字签名方案的算法)

[0191] 上面已描述了HFE函数F_t。接下来，将描述利用HFE函数F_t的数字签名方案(以下称为HFE签名方案)的具体算法。此外，使用HFE函数的PFDH签名方案(下面称为HFE+PFDH签名方案)在这里被描述为HFE签名方案的一个示例。

[0192] 《PFDH签名方案》

[0193] 首先，将描述PFDH签名方案的密钥生成算法Gen、签名生成算法Sig和签名验证算法Ver。PFDH签名方案的这些算法使用陷门单向函数$F_t:A\rightarrow B$和哈希函数$H:\{0,1\}^*\rightarrow B$。

[0194] (密钥生成算法Gen)

[0195] 密钥生成算法Gen以安全性参数1^λ作为输入，以签名密钥sk作为F_t的陷门t，以验证密钥pk作为F_t，并且计算$(sk,pk)((sk,pk)\leftarrow Gen(1^\lambda))$。

[0196] (签名生成算法Sig)

[0197] 签名生成算法Sig以消息M和签名密钥sk作为输入，并且通过下面的步骤1至步骤4来计算数字签名$\sigma(\sigma\leftarrow Sig(sk,M))$。

[0198] (步骤1)生成随机数r。

[0199] (步骤2)计算$y=H(M,r)\in B$。

[0200] (步骤3)利用陷门t来执行F_t的后向计算算法,并且计算满足$y=F_t(x)$的x。另外,当满足$y=F_t(x)$的x不存在时,返回步骤1。

[0201] (步骤4)输出数字签名$\sigma=(x,r)$。

[0202] (签名验证算法Ver)

[0203] 签名验证算法Ver以验证密钥$pk=F_t$、消息M和数字签名$\sigma=(x,r)$作为输入,并且通过下面的步骤1和步骤2来验证消息M的数字签名$\sigma(0/1\leftarrow Ver(pk,M,\sigma))$。

[0204] (步骤1)判断$F_t(x)$是否等于H(M,r)。

[0205] (步骤2)在$F_t(x)=H(M,r)$的情况中,输出1;在$F_t(x)\neq H(M,r)$的情况中,输出0。

[0206] 《HFE+PFDH签名方案》

[0207] 接下来,将描述HFE+PFDH签名方案的签名生成算法Sig和签名验证算法Ver。HFE+PFDH签名方案是使用HFE函数F_t的PFDH签名方案。另外,在HFE+PFDH签名方案中,签名密钥sk、HFE函数F_t的陷门S,T,a_{ij},b_i和c以及验证密钥pk被设置为F_t。

[0208] (签名生成算法Sig)

[0209] 签名生成算法Sig以消息M和签名密钥sk作为输入,并且通过下面的步骤1至步骤9来计算数字签名$\sigma(\sigma\leftarrow Sig(sk,M))$。

[0210] (步骤1)生成随机数r。

[0211] (步骤2)利用随机数r和消息M来计算哈希值$y\in K^n\leftarrow H(M,r)$。

[0212] (步骤3)通过将$y=(y_0,\cdots,y_{n-1})\in K^n$应用于变换T的逆变换$T^{-1}$来获得$y'=(y_0',\cdots,y_{n-1}')\in K^n$。

[0213] (步骤4)通过 φ^{-1}将$y'=(y_0',\cdots,y_{n-1}')\in K^n$变换为$Y'\in A$。

[0214] (步骤5)计算$X'\in\{Z\in A|f(Z)=Y'\}$的集合。

[0215] (步骤6)从集合$\{Z\in A|f(Z)=Y'\}$中选择一个元素X'。这里,在集合$\{Z\in A|f(Z)=Y'\}$为空集的情况中,返回步骤1的处理。

[0216] (步骤7)通过 φ将$X'\in A$变换为$x'=(x_0',\cdots,x_{n-1}')\in K^n$。

[0217] (步骤8)通过变换S将$x'\in K^n$变换为$x=(x_0,\cdots,x_{n-1})\in K^n$。

[0218] (步骤9)输出数字签名$\sigma=(x,r)$。

[0219] (签名验证算法Ver)

[0220] 签名验证算法Ver以验证密钥$pk=F_t$、消息M和数字签名$\sigma=(x,r)$作为输入,并且通过下面的步骤1至步骤3来验证消息M的数字签名σ的有效性$(0/1\leftarrow Ver(pk,M,\sigma))$。

[0221] (步骤1)通过利用包括在数字签名σ中的r和消息M来计算哈希值$y\leftarrow H(M,r)$。

[0222] (步骤2)通过将包括在数字签名σ中的$x\in K^n$代入HFE函数$F_t(x)$中来计算$y''=F_t(x)\in K^n$。

[0223] (步骤3)当$y=y''$时输出1,并且当$y\neq y''$时输出0。

[0224] 在前面,已描述了HFE数字签名方案(HFE+PFDH签名方案已在这里进行了说明)的每个算法的结构。如从上述算法结构可推测到的,根据HFE数字签名方案,如果与H(M,r)有关的$F_t=T*F*S$的前像被计算出,则数字签名σ将被伪造。即,HFE数字签名方案是依赖于求解由$H(M,r)=F_t(x)$表示的特殊多次多元联立方程式的难度的方案。难以评估求解这样的特殊多次多元联立方程式的难度。在实际中,据说在HFE数字签名方案中使用的多次多元联立方程式可以在准多项式时间或更少的时间中被求解出。

[0225] <2:第一实施例>

[0226] 这里,将描述本发明的第一实施例。与HFE数字签名方案一样,本实施例涉及采用求解多次多元联立方程式的难度作为安全性基础的公钥认证方案和数字签名方案。然而,与HFE数字签名方案等不同的是,本实施例提供了能够利用没有高效求解手段(陷门)的多次多元联立方程式来确保足够的安全性的公钥认证方案和数字签名方案。

[0227] [2-1:公钥认证方案的算法]

[0228] 首先,将参考图4描述本实施例的公钥认证方案(以下称为本方案)的算法。图4是用于描述本方案的算法的说明图。另外,本方案由密钥生成算法Gen、证明者算法P和验证者算法V构成。下面,将描述每个算法的内容。

[0229] (密钥生成算法Gen)

[0230] 密钥生成算法Gen生成在环K上定义的n个变量的m个多次多项式$f_1(x_1,\cdots,x_n),\cdots,f_m(x_1,\cdots,x_n)$以及向量$s=(s_1,\cdots,s_n)\in K^n$。接下来,该密钥生成算法Gen计算$y=(y_1,\cdots,y_m)\leftarrow(f_1(s),\cdots,f_m(s))$。然后,该密钥生成算法Gen设置$(f_1,\cdots,f_m,y)$作为公钥pk,并且设置s作为私钥。另外,在下面,n个变量的向量$(x_1,\cdots,x_n)$将被表达为x,并且n个变量的m个多次多项式$(f_1(x),\cdots,f_m(x))$将被表达为F(x)。

[0231] (证明者算法P、验证者算法V)

[0232] 接下来,将参考图4描述在交互式协议中通过证明者算法P和验证者算法V进行的处理。该交互式协议用于使得验证者证明“证明者知道满足y=F(s)的s”,而根本不会将关于s的信息泄露给验证者。另外,假设通过密钥生成算法Gen生成的公钥pk在证明者与验证者之间被共享。此外,假设通过密钥生成算法Gen生成的私钥sk由证明者秘密地管理。

[0233] 步骤1:

[0234] 首先,证明者算法P任意地选择一个数w。然后,证明者算法P通过将该数w应用于伪随机数生成器G_1来生成向量$r\in K^n$和数w'。即,证明者算法P计算$(r,w')\leftarrow G_1(w)$。接下来,证明者算法P通过将数w'应用于伪随 机数生成器G_2来生成n个变量的多项式$F'(x)=(f'_1(x),\cdots,f'_m(x))$。即,证明者算法P计算$F'\leftarrow G_2(w')$。

[0235] 步骤1(续):

[0236] 接下来,证明者算法P计算$z\leftarrow s-r$。该计算对应于通过向量r来掩蔽(mask)私钥s的操作。此外,证明者算法P计算$F''(x)\leftarrow F(x+r)+F'(x)$。该计算对应于通过多项式的集合F'(x)来掩蔽针对x的多项式的集合F(x+r)的操作。

[0237] 步骤1(续):

[0238] 接下来,证明者算法P生成F''(z)和z的哈希值c_1。即,证明者算法P计算$c_1\leftarrow H_1(F''(z),z)$。此外,证明者算法P生成数w'的哈希值$c_2$。即,证明者算法P计算$c_2\leftarrow H_2(w')$。此外,证明者算法P生成多项式的集合F''的哈希值c_3。即,证明者算法P计算$c_3\leftarrow H_3(F'')$。另外,所描述的$H_1(\cdots)$,$H_2(\cdots)$和$H_3(\cdots)$是哈希函数。此外,哈希值(c_1,c_2,c_3)是消息。

[0239] 在步骤1中生成的消息(c_1,c_2,c_3)被发送给验证者。

[0240] 步骤2:

[0241] 验证者算法V从三种验证模式中选择将被使用的验证模式。然后,验证者算法V向证明者发送指示所选验证模式的要求$d\in\{0,1,2\}$。

[0242] 步骤3:

[0243] 证明者算法P响应于从验证者接收的要求d来生成将被发送回验证者的信息σ。如果d＝0，则证明者算法P生成信息σ＝w。此外，如果d＝1，则证明者算法P生成信息σ＝(w’，z)。此外，如果d＝2，则证明者算法P生成信息σ＝(F”(x)，z)。以这种方式生成的信息σ通过证明者算法P被发送给验证者。

[0244] 步骤4：

[0245] 验证者算法V利用从证明者接收的信息σ来执行下面的验证处理。

[0246] 如果d＝0，则验证者算法V计算(r’，w”)←G_1(σ)。此外，验证者算法V计算F‴←G_2(w”)。然后，验证者算法V验证等式c_2＝H_2(w”)是否成立。此外，验证者算法V验证等式c_3＝H_3(F(x+r’)+F‴(x))是否成立。验证者算法V在所有验证都已成功的情况中输出指示认证成功的值1，并且在在验 证中出现了失败的情况中输出指示认证失败的值0。

[0247] 如果d＝1，则验证者算法V计算(w”，z’)←σ。此外，验证者算法V计算F‴←G_2(w”)。然后，验证者算法V验证等式c_1＝H_1(F‴(z’)，z’)是否成立。此外，验证者算法V验证等式c_2＝H_2(w”)是否成立。验证者算法V在所有验证都已成功的情况中输出指示认证成功的值1，并且在在验证中出现了失败的情况中输出指示认证失败的值0。

[0248] 如果d＝2，则验证者算法V计算(F⁗，z’)←σ。然后，验证者算法V验证等式c_1＝H_1(F⁗(z’)-y，z’))是否成立。此外，验证者算法V验证等式c_3＝H_3(F⁗)是否成立。验证者算法V在所有验证都已成功的情况中输出指示认证成功的值1，并且在在验证中出现了失败的情况中输出指示认证失败的值0。

[0249] (补充)

[0250] 此外，注意，在将上述步骤1中生成的消息(c_1，c_2，c_3)发送给验证者时，与私钥sk有关的信息、与r有关的信息以及与z有关的信息根本不会被泄露给验证者。还注意，在将上述步骤3中生成的信息σ发送给验证者时，与z有关的信息在d＝0的情况中根本不会被泄露给验证者，并且与r有关的信息在d＝1，2的情况中根本不会被泄露给验证者。

[0251] (本方案中的合理性)

[0252] 本方案的合理性通过如下方式来得到保证：如果证明者对来自验证者的所有要求d＝0，1，2响应以针对消息(c_1，c_2，c_3)的正确信息σ，则满足以下公式(8)和(9)的F⁗，F‴，r’和z’可从响应中被计算出。

[0253] [表达式7]

[0254] $$F''''(x)=F(x+r')+F'''(x) \quad \cdots(8)$$

[0255] $$F''''(z')-y=F'''(z') \quad \cdots(9)$$

[0256] 在这样的逻辑被保证的情况下，可以保证只要多次多元联立方程式不被求解出，就无法以高于2/3的概率执行伪造。即，为了正确地对验证者的所有要求d＝0，1，2作出响应，伪造者必须能够计算出满足上面公式(8)和(9)的F⁗，F‴，r’和z’。换言之，伪造者必须能够计算出满足F(s)＝y的s。此外，伪造者可能最多能够对验证者的要求d＝0，1，2中的两个 正确地作出响应。因此，伪造成功的概率将为2/3。上述交互式协议被执行足够的次数。因此，伪造成功的概率可被减小到可忽略的水平。

[0257] (修改)

[0258] 上述密钥生成算法Gen计算y←F(s)，并且将公钥设置为(F，y)。然而，密钥生成算法Gen可被配置为获得(y_1，…，y_m)←F(s)并计算(f_1^*(x)，…，f_m^*(x))←(f_1(x)-y_1，…，f_m(x)-

y_m),,并且将公钥设置为(f_1^*,…,f_m^*)。通过该修改,使得交互式协议能够在取y=0的情况下在证明者与验证者之间被执行。

[0259] 此外,上述证明者算法P从F"(z)和z生成消息c_1,但是由于具有关系F"(z)=F'(z),因此当从F'(z)和z生成消息c_1时也可以配置类似的交互式协议。此外,证明者算法P可以分开计算F"(z)的哈希值和z的哈希值,并且可以将每个作为消息发送给验证者。

[0260] 此外,上述证明者算法P通过将伪随机数生成器G_1应用于数w来生成向量r和数w'。然后,证明者算法P通过将数w'应用于伪随机数生成器G_2.来生成n个变量的多项式F'(x)。然而,证明者算法P可以在最初计算出w=(r,F'),并且可以对G_1进行恒等映射。此外,在此情况中,数w不必被应用于G_1。另外,这也同样适用于G_2。

[0261] 此外,上述交互式协议采取(F,y)作为公钥。该F是不依赖于私钥sk的参数。因此,也可以使得该参数F成为整个系统共有的参数,而不是被提供给每个证明者。在此情况中,为各个证明者设置的公钥可以仅为y,并且公钥的大小可被减小。

[0262] 此外,上述交互式协议采取(f_1,…,f_m,y)作为公钥,但是F=(f_1,…,f_m,)是可被适当选择的参数。因此,证明者和验证者可以不是使(f_1,…,f_m,)保持原样,而是准备随机数的种子w_{pk}并且可以利用伪随机数生成器G^*来计算F←G^*(w_{pk})。在此情况中,公钥将为(w_{pk},y),并且可以使公钥的大小比将(F,y)公开作为公钥时小。

[0263] 在上述方案中,c_1,c_2和c_3是利用哈希函数H_1,H_2和H_3计算的,但是还可以使用承诺方案COM来取代哈希函数。承诺函数COM是以字符串S和随机数ρ作为自变数的函数。承诺函数的示例包括由Shai Halevi和 Silvio Micali在1996年的国际会议CRYPTO中介绍的方案等。

[0264] 在使用该承诺函数的情况中,在计算c_1,c_2和c_3之前准备随机数ρ_1,ρ_2和ρ_3,并且通过应用承诺函数COM(·,ρ_1),COM(·,ρ_2)和COM(·,ρ_3)取代哈希函数H_1(·),H_2(·)和H_3(·)来生成c_1,c_2和c_3。此外,在此修改中,用于生成将由验证单元计算出C_i所需的ρ_1被包括在响应信息中。另外,该修改方案可以应用于下面描述的所有方案。

[0265] 在前面,已描述了根据本方案的基本算法结构。

[0266] [2-2:扩展算法]

[0267] 接下来,将参考图5描述作为本方案的扩展(下面称为扩展方案)的公钥认证方案的算法。图5是用于描述基于扩展方案的交互式协议的流程的说明图。该扩展方案是用于将在第一次传递中被发送的消息(c_1,c_2和c_3)变换为一个哈希值c并将其发送给验证者的方案。此外,根据该扩展方案,交互式协议被配置为在第一次传递中发送哈希值c,并且因此未从在第三次传递中发送的信息σ中被恢复的消息与信息σ一起被发送给验证者。通过这样的扩展,将在交互式协议中被发送给验证者的哈希值的数目可被减少,并且将被传输的数据的大小可被减小。下面,将详细描述扩展方案的每个算法的内容。

[0268] (密钥生成算法Gen)

[0269] 密钥生成算法Gen生成在环K上定义的n个变量的m个多次多项式f_1(x_1,…,x_n),…,f_m(x_1,…,x_n)以及向量s=(s_1,…,s_n)∈K^n。接下来,该密钥生成算法Gen计算y=(y_1,…,y_m)←(f_1(s),…,f_m(s))。然后,该密钥生成算法Gen设置(f_1,…,f_m,y)作为公钥pk,并且设置s作为私钥。另外,在下面,n个变量的向量(x_1,…,x_n)将被表达为x,并且n个变量的m个多次多项式(f_1(x),…,f_m(x))将被表达为F(x)。

[0270] （证明者算法P、验证者算法V）

[0271] 接下来，将参考图5描述在交互式协议中通过证明者算法P和验证者算法V进行的处理。该交互式协议用于使得验证者证明“证明者知道满足y＝F(s)的s”，而根本不会将关于s的信息泄露给验证者。另外，假设通 过密钥生成算法Gen生成的公钥pk在证明者与验证者之间被共享。此外，假设通过密钥生成算法Gen生成的私钥sk由证明者秘密地管理。

[0272] 步骤1：

[0273] 首先，证明者算法P任意地选择一个数w。然后，证明者算法P通过将该数w应用于伪随机数生成器G_1来生成向量$r\in K^n$和数w’。即，证明者算法P计算$(r,w')\leftarrow G_1(w)$。接下来，证明者算法P通过将数w’应用于伪随机数生成器G_2来生成n个变量的多项式$F'(x)=(f'_1(x),\cdots,f'_m(x))$。即，证明者算法P计算$F'\leftarrow G_2(w')$。

[0274] 步骤1(续)：

[0275] 接下来，证明者算法P计算$z\leftarrow s-r$。该计算对应于通过向量r来掩蔽私钥s的操作。此外，证明者算法P计算$F''(x)\leftarrow F(x+r)+F'(x)$。该计算对应于通过多项式的集合F’(x)来掩蔽针对x的多项式的集合F(x+r)的操作。

[0276] 步骤1(续)：

[0277] 接下来，证明者算法P生成F”(z)和z的哈希值c_1。即，证明者算法P计算$c_1\leftarrow H_1(F''(z),z)$。此外，证明者算法P生成数w’的哈希值$c_2$。即，证明者算法P计算$c_2\leftarrow H_2(w')$。此外，证明者算法P生成多项式的集合F”的哈希值$c_3$。即，证明者算法P计算$c_3\leftarrow H_3(F'')$。另外，所描述的$H_1(\cdots)$，$H_2(\cdots)$和$H_3(\cdots)$是哈希函数。此外，哈希值$(c_1,c_2,c_3)$是消息。

[0278] 步骤1(续)：

[0279] 在扩展方案的情况中，证明者算法P通过将消息(c_1,c_2,c_3)应用于哈希函数H来生成哈希值c。然后，证明者算法P将所生成的哈希值c发送给验证者。

[0280] 步骤2：

[0281] 验证者算法V从三种验证模式中选择将被使用的验证模式。然后，验证者算法V向证明者发送指示所选验证模式的要求$d\in\{0,1,2\}$。

[0282] 步骤3：

[0283] 证明者算法P响应于从验证者接收的要求d来生成将被发送回验证者的信息σ。如果d＝0，则证明者算法P生成信息$(\sigma,c^*)=(w,c_1)$。此外，如果d＝1，则证明者算法P生成信息$(\sigma,c^*)=((w',z),c_3)$。此外，如果d＝ 2，则证明者算法P生成信息$(\sigma,c^*)=((F'',z),c_2)$。以这种方式生成的信息$(\sigma,c^*)$通过证明者算法P发送给验证者。

[0284] 步骤4：

[0285] 验证者算法V利用从证明者接收的信息(σ,c^*)来执行下面的验证处理。

[0286] 如果d＝0，则验证者算法V计算$(r',w'')\leftarrow G_1(\sigma)$。此外，验证者算法V计算$F'''\leftarrow G_2(w'')$。然后，验证者算法V计算$c'_2=H_2(w'')$。然后，验证者算法V计算$c'_3=H_3(F(x+r')+F'''(x))$。然后，验证者算法V验证等式$c=H(c^*,c'_2,c'_3)$是否成立。验证者算法V在验证已成功的情况中输出指示认证成功的值1，并且在在验证中出现了失败的情况中输出指示认证失败的值0。

[0287] 如果d＝1，则验证者算法V计算$(w'',z')\leftarrow\sigma$。接下来，验证者算法V计算$F'''\leftarrow G_2(w'')$。然后，验证者算法V计算$c'_1=H_1(F'''(z'),z')$。然后，验证者算法V计算$c'_2=H_2(w'')$。

然后，验证者算法V验证等式$c=H(c'_1,c'_2,c^*)$是否成立。验证者算法V在验证已成功的情况中输出指示认证成功的值1，并且在在验证中出现了失败的情况中输出指示认证失败的值0。

[0288] 如果$d=2$，则验证者算法V计算$(F'',z')\leftarrow\sigma$。然后，验证者算法V计算$c'_1=H_1(F''(z')-y,z')$。然后，验证者算法V计算$c'_3=H_3(F'')$。然后，验证者算法V验证等式$c=H(c'_1,c^*,c'_3)$是否成立。验证者算法V在验证已成功的情况中输出指示认证成功的值1，并且在在验证中出现了失败的情况中输出指示认证失败的值0。

[0289] 在前面，已描述了扩展方案的交互式协议中的每个算法的处理。通过这样扩展，将在交互式协议中被发送给验证者的哈希值的数目可被减少，并且将被传输的数据的大小可被减小。

[0290] [2-3：并行算法]

[0291] 现在，如已经描述的，当采用根据本方案或扩展方案的交互式协议时，伪造成功的概率可被减小至2/3或更小。因此，如果该交互式协议被执行两次，则伪造成功的概率可被减小至$(2/3)^2$或更小。以相同方式，如果 该交互式协议被执行N次，伪造成功的概率变为$(2/3)^N$，并且如果N是足够大的数(例如，$N=140$)，则伪造成功的概率被减少到可忽略的水平。例如，将根据本方案的交互式协议并行地执行N次的算法在图6中示出。下面，将参考图6描述将交互式协议并行地执行N次的每个算法的内容。

[0292] (密钥生成算法Gen)

[0293] 密钥生成算法Gen生成在环K上定义的n个变量的m个多次多项式$f_1(x_1,\cdots,x_n),\cdots,f_m(x_1,\cdots,x_n)$以及向量$s=(s_1,\cdots,s_n)\in K^n$。接下来，该密钥生成算法Gen计算$y=(y_1,\cdots,y_m)\leftarrow(f_1(s),\cdots,f_m(s))$。然后，该密钥生成算法Gen设置$(f_1,\cdots,f_m,y)$作为公钥pk，并且设置s作为私钥。另外，在下面，n个变量的向量$(x_1,\cdots,x_n)$将被表达为x，并且n个变量的m个多次多项式$(f_1(x),\cdots,f_m(x))$将被表达为F(x)。

[0294] (证明者算法P、验证者算法V)

[0295] 接下来，将参考图6描述在交互式协议中通过证明者算法P和验证者算法V进行的处理。该交互式协议用于使得验证者证明“证明者知道满足$y=F(s)$的s”，而根本不会将关于s的信息泄露给验证者。另外，假设通过密钥生成算法Gen生成的公钥pk在证明者与验证者之间被共享。此外，假设通过密钥生成算法Gen生成的私钥sk由证明者秘密地管理。

[0296] 步骤1：

[0297] 首先，证明者算法P针对$i=1$至N执行下面的处理(1)至处理(8)。(处理1)证明者算法P任意地选择一个数w_i。(处理2)证明者算法P通过将该数w_i应用于伪随机数生成器G_1来生成向量$r_i\in K^n$和数w'_i。即，证明者算法P计算$(r_i,w'_i)\leftarrow G_1(w_i)$。(处理3)证明者算法P通过将数$w'_i$应用于伪随机数生成器$G_2$来生成n个变量的多项式的集合$F'_i(x)$。即，证明者算法P计算$F'_i\leftarrow G_2(w'_i)$。

[0298] 步骤1(续)：

[0299] (处理4)证明者算法P计算$z_i\leftarrow s-r_i$。该计算对应于通过向量r_i来掩蔽私钥s的操作。(处理5)证明者算法P计算$F''_i(x)\leftarrow F(x+r_i)+F'_i(x)$。该计算对应于通过多项式的集合$F'_i(x)$来掩蔽针对x的多项式的集合$F(x+r_i)$的操作。

[0300] 步骤1(续)：

[0301] (处理6)证明者算法P生成F"$_i$(z_i)和z_i的哈希值$c_{1,i}$。即，证明者算法P计算$c_{1,i}$←H_1(F"$_i$(z_i)，z_i)。(处理7)证明者算法P生成数w'$_i$的哈希值$c_{2,i}$。即，证明者算法P计算$c_{2,i}$←H_2(w'$_i$)。(处理8)证明者算法P生成多项式的集合F"$_i$的哈希值$c_{3,i}$。即，证明者算法P计算$c_{3,i}$←H_3(F"$_i$)。另外，所描述的H_1(…)，H_2(…)和H_3(…)是哈希函数。此外，哈希值($c_{1,i}$，$c_{2,i}$，$c_{3,i}$)是消息。

[0302] 当针对i＝1至N执行了上述(处理1)至(处理8)之后，在步骤1中生成的消息($c_{1,i}$，$c_{2,i}$，$c_{3,i}$)(i＝1至N)被发送给验证者。

[0303] 步骤2：

[0304] 验证者算法V针对i＝1至N的每个从三种验证模式中选择将被使用的验证模式。然后，验证者算法V向证明者发送指示所选验证模式的要求d_i∈{0，1，2}(i＝1至N)。

[0305] 步骤3：

[0306] 证明者算法P响应于从验证者接收的要求d_i来生成将被发送回验证者的信息σ_i。这里，证明者算法P针对i＝1至N来执行下面的(处理1)至(处理3)。(处理1)如果d_i＝0，则证明者算法P生成信息σ_i＝w_i。(处理2)如果d_i＝1，则证明者算法P生成信息σ_i＝(w'$_i$，z_i)。(处理3)如果d_i＝2，则证明者算法P生成信息σ_i＝(F"$_i$，z_i)。当上述(处理1)至(处理3)的决定和处理已被执行之后，信息σ_i(i＝1至N)通过证明者算法P被发送给验证者。

[0307] 步骤4：

[0308] 验证者算法V利用从证明者接收的信息σ_i(i＝1至N)来执行下面的验证处理。另外，下面的处理针对i＝1至N被执行。

[0309] 如果d_i＝0，则验证者算法V计算(r'$_i$，w"$_i$)←G_1(σ_i)。此外，验证者算法V计算F‴$_i$←G_2(w"$_i$)。然后，验证者算法V验证等式$c_{2,i}$＝H_2(w"$_i$)是否成立。此外，验证者算法V验证等式$c_{3,i}$＝H_3(F(x+r'$_i$)+F‴$_i$(x))是否成立。验证者算法V在所有验证都已成功的情况中输出指示认证成功的值1，并且在在验证中出现了失败的情况中输出指示认证失败的值0。

[0310] 如果d_i＝1，则验证者算法V计算(w"$_i$，z'$_i$)←σ_i。此外，验证者算法V计算F‴$_i$←G_2(w"$_i$)。然后，验证者算法V验证等式$c_{1,i}$＝H_1(F‴$_i$(z'$_i$)，z'$_i$)是否成立。此外，验证者算法V验证等式$c_{2,i}$＝H_2(w"$_i$)是否成立。验证者算法V在所有验证都已成功的情况中输出指示认证成功的值1，并且在在验证中出现了失败的情况中输出指示认证失败的值0。

[0311] 如果d_i＝2，则验证者算法V计算(F⁗$_i$，z'$_i$)←σ_i。然后，验证者算法V验证等式$c_{1,i}$＝H_1(F⁗$_i$(z'$_i$)-y，z'$_i$)是否成立。此外，验证者算法V验证等式$c_{3,i}$＝H_3(F⁗$_i$(x))是否成立。验证者算法V在所有验证都已成功的情况中输出指示认证成功的值1，并且在在验证中出现了失败的情况中输出指示认证失败的值0。

[0312] 在前面，已描述了并行地执行本方案的交互式协议的方法。如上所述，通过重复地执行本方案的交互式协议，伪造成功的概率可被减小到可忽略的水平。

[0313] 另外，在上述步骤1之后，取代发送($c_{1,1}$，$c_{1,2}$，$c_{1,3}$，…，$c_{N,1}$，$c_{N,2}$，$c_{N,3}$)给验证者，它们的哈希值H($c_{1,1}$，$c_{1,2}$，$c_{1,3}$，…，$c_{N,1}$，$c_{N,2}$，$c_{N,3}$)可以总体地被发送。当采用这种配置时，在第一次传递中发送的消息将仅是一个哈希值，并且通信量可被大幅减少。此外，该哈希值以及未从发送自证明者的响应中被恢复为挑战的哈希值将与该响应一起从证明者被发送。根据这种配置，在并行重复N次的情况中，将被发送的信息的个数可被减少2N-1个。

[0314] (参数设置的合适方法)

[0315] 现在，根据本实施例的交互式协议保证了针对被动攻击的安全性。然而，在采用并行地重复执行该交互式协议的上述方法的情况中，为了保证针对主动攻击的安全性得到绝对保证，必须满足如下所述的条件。

[0316] 上述交互式协议是证明者利用一对密钥(公钥y、私钥s)向验证者证明“对于y，该证明者知道满足y＝F(s)的s”的方案。因此，如果通过验证而被接受的交互被执行，则不能否认这样的可能性，即“证明者在交互时使用了该s”的信息将被验证者所知。此外，对于上面的F，抗冲突能力(collision resistance)得不到保证。因此，对于在无条件的情形下并行 地重复执行上述交互式协议的情况，难以证明针对主动攻击的安全性将得到绝对保证。

[0317] 因此，本申请的发明人已调研出了一种方法，该方法即使在通过验证而被接受的交互被执行时也可防止“证明者在交互时使用了该s”的信息被验证者所知。于是，本申请的发明人已想出了即使在并行地重复执行上述交互式协议的情况中也能够保证针对主动攻击的安全性的方法。该方法用于将用作公钥的n个变量的多次多项式$f_1,\cdots,f_m$的数目m设置为充分小于变量的数目n的值。例如，m和n被设置为使得$2^{m-n}<<1$(例如，当n＝160并且m＝80时，$2^{-80}<<1$)。

[0318] 在采取求解多次多元方程式的难度作为安全性基础的上述方案中，即使私钥s_1和与该私钥相对应的公钥pk被给出，也难以生成与该公钥pk相对应的另一私钥s_2。因此，如果保证了对于公钥pk存在两个以上的私钥s，则即使当通过验证而被接受的交换被执行时，也可以防止“证明者在交互时使用了该s”的信息被验证者所知。即，如果该保证成立，则即使在并行地重复执行交互式协议的情况中，针对主动攻击的安全性也可以得到保证。

[0319] 参考图28，考虑由n个变量的m个多次多项式形成的函数$F:K^n\rightarrow K^m$(其中，n＞m)，不具有第二前像的定义域中的元素数目最大仅为$|K|^{m-1}$。因此，如果使$|K|^{m-n}$充分小，则不具有第二前像的定义域中的元素被选择的可能性可以被降低到可忽略的水平。即，如果n个变量的多次多项式$f_1,\cdots,f_m$的数目m被设置为充分小于变量的数目n的值，则可以保证对于公钥pk存在两个以上私钥s。结果，同样在通过验证而被接受的交互被执行的情况中，可以防止“证明者在交互时使用了该s”的信息被验证者所知，并且即使在并行地重复执行交互式协议的情况中，针对主动攻击的安全性也将得到保证。

[0320] 如上所述，通过采用将n个变量的多次多项式$f_1,\cdots,f_m$的数目m设置为小于变量的数目n的值(n＞m；优选地，$2^{m-n}<<1$)的方法，可以提高并行地重复执行交互式协议的情况的安全性。

[0321] [2-4：非交互式算法]

[0322] 在前面，已描述了3次传递公钥认证方案。然而，由于根据本方案，在第二次传递中从验证者发送给证明者的信息仅是指示验证模式的要求d(实际上，在许多情况中，仅随机数被使用)，因此可以修改为1次传递公钥认证方案(下面称为非交互式方案)。另外，根据非交互式方案的每个算法的内容在图7中示出。下面，将参考图7描述根据非交互式方案的每个算法的内容。

[0323] (密钥生成算法Gen)

[0324] 密钥生成算法Gen生成在环K上定义的n个变量的m个多次多项式$f_1(x_1,\cdots,x_n),\cdots,f_m(x_1,\cdots,x_n)$以及向量$s=(s_1,\cdots,s_n)\in K^n$。接下来，该密钥生成算法Gen计算$y=(y_1,\cdots,y_m)\leftarrow(f_1(s),\cdots,f_m(s))$。然后，该密钥生成算法Gen设置$(f_1,\cdots,f_m,y)$作为公钥pk，并且设置s

作为私钥。另外，在下面，n个变量的向量$(x_1,\cdots,x_n)$将被表达为x，并且n个变量的m个多次多项式$(f_1(x),\cdots,f_m(x))$将被表达为F(x)。

[0325] (证明者算法P、验证者算法V)

[0326] 接下来，将参考图7描述在交互式协议中通过证明者算法P和验证者算法V进行的处理。该交互式协议用于使得验证者证明“证明者知道满足y＝F(s)的s”，而根本不会将关于s的信息泄露给验证者。另外，假设通过密钥生成算法Gen生成的公钥pk在证明者与验证者之间被共享。此外，假设通过密钥生成算法Gen生成的私钥sk由证明者秘密地管理。

[0327] 步骤1：

[0328] 首先，证明者算法P针对i＝1至N执行下面的处理(1)至处理(8)。(处理1)证明者算法P任意地选择一个数w_i。(处理2)证明者算法P通过将该数w_i应用于伪随机数生成器G_1来生成向量$r_i\in K^n$和数w'_i。即，证明者算法P计算$(r_i,w'_i)\leftarrow G_1(w_i)$。(处理3)证明者算法P通过将数$w'_i$应用于伪随机数生成器$G_2$来生成n个变量的多项式的集合$F'_i(x)$。即，证明者算法P计算$F'_i\leftarrow G_2(w'_i)$。

[0329] 步骤1(续)：

[0330] (处理4)证明者算法P计算$z_i\leftarrow s-r_i$。该计算对应于通过向量r_i来掩 蔽私钥s的操作。(处理5)证明者算法P计算$F''_i(x)\leftarrow F(x+r_i)+F'_i(x)$。该计算对应于通过多项式的集合$F'_i(x)$来掩蔽针对x的多项式的集合$F(x+r_i)$的操作。

[0331] 步骤1(续)：

[0332] (处理6)证明者算法P生成$F''_i(z_i)$和z_i的哈希值$c_{1,i}$。即，证明者算法P计算$c_{1,i}\leftarrow H_1(F''_i(z_i),z_i)$。(处理7)证明者算法P生成数$w'_i$的哈希值$c_{2,i}$。即，证明者算法P计算$c_{2,i}\leftarrow H_2(w'_i)$。(处理8)证明者算法P生成多项式的集合$F''_i$的哈希值$c_{3,i}$。即，证明者算法P计算$c_{3,i}\leftarrow H_3(F''_i)$。另外，所描述的$H_1(\cdots)$，$H_2(\cdots)$和$H_3(\cdots)$是哈希函数。此外，哈希值$(c_{1,i},c_{2,i},c_{3,i})$是消息。

[0333] 步骤2：

[0334] 接下来，证明者算法P选择随机数R。然后，证明者算法P对于i＝1至N，通过将随机数R和在步骤1中生成的消息$(c_{1,i},c_{2,i},c_{3,i})$应用于哈希函数H来生成$d=(d_1,\cdots,d_N)$。

[0335] 步骤3：

[0336] 接下来，证明者算法P根据所生成的d_i来生成将被发送给验证者的信息σ_i。这里，证明者算法P针对i＝1至N来执行下面的(处理1)至(处理3)。(处理1)如果$d_i=0$，则证明者算法P生成信息$\sigma_i=w_i$。(处理2)如果$d_i=1$，则证明者算法P生成信息$\sigma_i=(w'_i,z_i)$。(处理3)如果$d_i=2$，则证明者算法P生成信息$\sigma_i=(F''_i,z_i)$。当上述(处理1)至(处理3)的决定和处理已被执行之后，随机数R、消息$(c_{1,i},c_{2,i},c_{3,i})$和信息$\sigma_i$(i＝1至N)通过证明者算法P被发送给验证者。

[0337] 步骤4：

[0338] 验证者算法V首先通过将接收自证明者的随机数R、消息$(c_{1,i},c_{2,i},c_{3,i})$和信息$\sigma_i$(i＝1至N)应用于哈希函数H来生成$d=(d_1,\cdots,d_N)$。然后，验证者算法V利用信息$\sigma_i$(i＝1至N)来执行下面的验证处理。另外，下面的处理针对i＝1至N被执行。

[0339] 如果$d_i=0$，则验证者算法V计算$(r'_i,w''_i)\leftarrow G_1(\sigma_i)$。此外，验证者算法V计算$F''_i\leftarrow G_2(w''_i)$。然后，验证者算法V验证等式$c_{2,i}=H_2(w''_i)$是否 成立。此外，验证者算法V验证等式

$c_{3,i}=H_3(F(x+r'_i)+F''_i(x))$是否成立。验证者算法V在所有验证都已成功的情况中输出指示认证成功的值1，并且在在验证中出现了失败的情况中输出指示认证失败的值0。

[0340] 如果$d_i=1$，则验证者算法V计算$(w''_i,z'_i)\leftarrow\sigma_i$。此外，验证者算法V计算$F''_i\leftarrow G_2(w''_i)$。然后，验证者算法V验证等式$c_{1,i}=H_1(F''_i(z'_i),z'_i)$是否成立。此外，验证者算法V验证等式$c_{2,i}=H_2(w''_i)$是否成立。验证者算法V在所有验证都已成功的情况中输出指示认证成功的值1，并且在在验证中出现了失败的情况中输出指示认证失败的值0。

[0341] 如果$d_i=2$，则验证者算法V计算$(F''_i,z'_i)\leftarrow\sigma_i$。然后，验证者算法V验证等式$c_{1,i}=H_1(F''_i(z'_i)-y,z'_i)$是否成立。此外，验证者算法V验证等式$c_{3,i}=H_3(F''_i)$是否成立。验证者算法V在所有验证都已成功的情况中输出指示认证成功的值1，并且在在验证中出现了失败的情况中输出指示认证失败的值0。

[0342] 在前面，已描述了根据非交互式方案的每个算法的内容。如上所述，根据非交互式方案，取代验证者向证明者发送随机数d以选择验证模式，证明者算法P通过利用消息$(c_{1,i},c_{2,i},c_{3,i})$来生成d。如果理想的哈希函数H被假定，则哈希值d将随机地表现，因此方便于证明者的哈希值d将不会出现。因此，即使在如上所述的修改被执行时，也将确保足够的安全性。另外，这样的修改同样可以应用于扩展方案等。

[0343] [2-5：修改为数字签名方案]

[0344] 这里，将描述将本方案修改为数字签名方案的方法。另外，出于简化考虑，这里将描述将非交互式方案修改为数字签名方案的方法。可以明白，当上述非交互式方案的证明者和验证者对应于数字签名方案的签名者和验证者时，对于数字签名方案的模型来说存在的类似性在于只有证明者可以取信验证者。考虑到该概念，将详细描述基于非交互式方案的数字签名方案的算法结构。

[0345] (密钥生成算法Gen)

[0346] 该密钥生成算法Gen生成在环K上定义的n个变量的m个多次多项 式$f_1(x_1,\cdots,x_n),\cdots,f_m(x_1,\cdots,x_n)$以及向量$s=(s_1,\cdots,s_n)\in K^n$。接下来，该密钥生成算法Gen计算$y=(y_1,\cdots,y_m)\leftarrow(f_1(s),\cdots,f_m(s))$。然后，该密钥生成算法Gen设置$(f_1,\cdots,f_m,y)$作为公钥pk，并且设置s作为私钥。另外，在下面，n个变量的向量$(x_1,\cdots,x_n)$将被表达为x，并且n个变量的m个多次多项式$(f_1(x),\cdots,f_m(x))$将被表达为F(x)。

[0347] (签名生成算法Sig)

[0348] 该签名生成算法Sig针对i＝1至N来执行下面的(处理1)至(处理15)。另外，假设签名密钥$sk=(s_1,\cdots,s_N)$和文档M被输入签名生成算法Sig中。

[0349] (处理1)签名生成算法Sig任意地选择数w_i。(处理2)签名生成算法Sig通过将该数w_i应用于伪随机数生成器G_1来生成向量$r_i\in K^n$和数w'_i。即，签名生成算法Sig计算$(r_i,w'_i)\leftarrow G_1(w_i)$。(处理3)签名生成算法Sig通过将数$w'_i$应用于伪随机数生成器$G_2$来生成n个变量的m个多项式$F'_i(x)=(f'_{1,i}(x),\cdots,f'_{m,i}(x)))$。即，签名生成算法Sig计算$F'_i\leftarrow G_2(w'_i)$。

[0350] (处理4)签名生成算法Sig计算$z_i\leftarrow s_i-r_i$。该计算对应于通过向量r_i来掩蔽私钥s_i的操作。(处理5)签名生成算法Sig计算$F''_i(x)\leftarrow F(x+r_i)+F'_i(x)$。该计算对应于通过多项式的集合$F'_i(x)$来掩蔽针对x的多项式的集合$F(x+r_i)$的操作。

[0351] (处理6)签名生成算法Sig生成$F''_i(z_i)$和z_i的哈希值$c_{1,i}$。即，签名生成算法Sig计算$c_{1,i}\leftarrow H_1(F''_i(z_i),z_i)$。(处理7)签名生成算法Sig生成数$w'_i$的哈希值$c_{2,i}$。即，签名生成算

法Sig计算$c_{2,i}\leftarrow H_2(w'_i)$。(处理8)签名生成算法Sig生成多项式的集合$F''_i$的哈希值$c_{3,i}$。即，签名生成算法Sig计算$c_{3,i}\leftarrow H_3(F''_i)$。另外，所描述的$H_1(\cdots)$，$H_2(\cdots)$和$H_3(\cdots)$是哈希函数。

[0352] (处理9)签名生成算法Sig选择随机数R。(处理10)签名生成算法Sig对于i＝1至N，通过将文档M、随机数R和哈希值($c_{1,i}$，$c_{2,i}$，$c_{3,i}$)应用于哈希函数H来生成$d=(d_1,\cdots,d_N)$。即，签名生成算法Sig计算$d=(d_1,\cdots,d_N)\leftarrow H(R,M,c_{1,1},\cdots,c_{3,N})$。(处理11)签名生成算法Sig根据所生成的$d_i$来生成信息$\sigma_1$。

[0353] 然后，该签名生成算法Sig针对i＝1至N来执行下面的(处理12) 至(处理14)。(处理12)如果$d_i=0$，则签名生成算法Sig生成信息$\sigma_i=w_i$。(处理13)如果$d_i=1$，则签名生成算法Sig生成信息$\sigma_i=(w'_i,z_i)$。(处理14)如果$d_i=2$，则签名生成算法Sig生成信息$\sigma_i=(F''_i,z_i)$。

[0354] (处理15)在针对i＝1至N执行了上面的(处理12)至(处理14)的决定和处理之后，签名生成算法Sig输出包括随机数R、消息($c_{1,i}$，$c_{2,i}$，$c_{3,i}$)和信息σ_i(i＝1至N)的数字签名$\sigma=(R,c_{1,i},c_{2,i},c_{3,i},\sigma_1,\cdots,\sigma_N)$。

[0355] (签名验证算法Ver)

[0356] 如果对于i＝1至N下面的验证都通过了，则签名验证算法Ver接受该数字签名σ，并且如果即使一个验证未通过，则签名验证算法Ver也否决该数字签名σ。另外，假设数字签名σ和文档M被输入签名验证算法Ver中。首先，该签名验证算法Ver计算$d=(d_1,\cdots,d_N)\leftarrow H(R,M,c_{1,1},\cdots,c_{3,N})$。接下来，该签名验证算法Ver针对i＝1至N来执行下面的(验证1)至(验证3)。

[0357] (验证1)如果$d_i=0$，则签名验证算法Ver计算$(r'_i,w''_i)\leftarrow G_1(\sigma_i)$。接下来，签名验证算法Ver计算$F'''_i\leftarrow G_2(w''_i)$。然后，签名验证算法Ver验证等式$c_{2,i}=H_2(w''_i)$是否成立。此外，签名验证算法Ver验证等式$c_{3,i}=H_3(F(x+r'_i)+F'''_i(x))$是否成立。签名验证算法Ver在所有验证都已成功的情况中输出指示接受该数字签名σ的值1，并且在在验证中出现了失败的情况中输出指示否决该数字签名σ的值0。

[0358] (验证2)如果$d_i=1$，则签名验证算法Ver计算$(w''_i,z'_i)\leftarrow\sigma_i$。接下来，签名验证算法Ver计算$F'''_i\leftarrow G_2(w''_i)$。然后，签名验证算法Ver验证等式$c_{1,i}=H_1(F'''_i(z'_i),z'_i)$是否成立。此外，签名验证算法Ver验证等式$c_{2,i}=H_2(w''_i)$是否成立。签名验证算法Ver在所有验证都已成功的情况中输出指示接受该数字签名σ的值1，并且在在验证中出现了失败的情况中输出指示否决该数字签名σ的值0。

[0359] (验证3)如果$d_i=2$，则签名验证算法Ver计算$(F''''_i,z'_i)\leftarrow\sigma_i$。然后，签名验证算法Ver验证等式$c_{1,i}=H_1(F''''_i(z'_i)-y,z'_i)$是否成立。此外，签名验证算法Ver验证等式$c_{3,i}=H_3(F''''_i)$是否成立。签名验证算法Ver在所有验证都已成功的情况中输出指示接受该数字签名σ的值1，并且在在验 证中出现了失败的情况中输出指示否决该数字签名σ的值0。

[0360] 在前面，已描述了基于本方案的数字签名方案的每个算法结构。基于上述扩展方案的数字签名方案可以同样地被构成。

[0361] [2-6:具体示例]

[0362] 接下来，将参考图8描述在执行本方案时可假定的具体算法结构。图8是用于描述本方案的具体示例的说明图。

[0363] (密钥生成算法Gen)

[0364] 密钥生成算法Gen生成在环K上定义的n个变量的m个二次多项式$f_1(x_1,\cdots,x_n),\cdots,f_m(x_1,\cdots,x_n)$以及向量$s=(s_1,\cdots,s_n)\in K^n$。接下来，该密钥生成算法Gen计算$y=(y_1,\cdots,y_m)\leftarrow(f_1(s),\cdots,f_m(s))$。然后，该密钥生成算法Gen设置$(f_1,\cdots,f_m,y)$作为公钥pk，并且设置s作为私钥。另外，在下面，n个变量的向量$(x_1,\cdots,x_n)$将被表达为x，并且n个变量的m个二次多项式$(f_1(x),\cdots,f_m(x))$将被表达为F(x)。此外，n个变量的二次多项式$f_i(x_1,\cdots,x_n)$被表示为如下的公式(10)。

[0365] [表达式8]

[0366] $$f_i(x_1,\cdots,x_n)=\sum_{j,k}a_{i,j,k}x_jx_k+\sum_j b_{i,j}x_j+c_i \quad \cdots(10)$$

[0367] (证明者算法P、验证者算法V)

[0368] 接下来，将参考图8描述在交互式协议中通过证明者算法P和验证者算法V进行的处理。该交互式协议用于使得验证者证明“证明者知道满足y=F(s)的s”，而根本不会将关于s的信息泄露给验证者。另外，假设通过密钥生成算法Gen生成的公钥pk在证明者与验证者之间被共享。此外，假设通过密钥生成算法Gen生成的私钥sk由证明者秘密地管理。

[0369] 步骤1：

[0370] 首先，证明者算法P任意地选择一个数w。然后，证明者算法P通过将该数w应用于伪随机数生成器G_1来生成向量$r\in K^n$和数w’。即，证明者算法P计算$(r,w')\leftarrow G_1(w)$。接下来，证明者算法P通过将数w’应用于伪随 机数生成器G_2来生成m个一次多项式的集合$f'_1(x_1,\cdots,x_n),\cdots,f'_m(x_1,\cdots,x_n)$。即，证明者算法P计算$(f'_1,\cdots,f'_m)\leftarrow G_2(w')$。另外，一次多项式$f'_1(x_1,\cdots,x_n)$被表示为如下的公式(11)。此外，m个一次多项式的集合$(f'_1(x),\cdots,f'_m(x))$将被表达为F’(x)。

[0371] [表达式9]

[0372] $$f'_i(x_1,\cdots,x_n)=\sum_j b'_{i,j}x_j+c'_i \quad \cdots(11)$$

[0373] 步骤1(续)：

[0374] 接下来，证明者算法P计算$z\leftarrow s-r$。该计算对应于通过向量r来掩蔽私钥s的操作。此外，证明者算法P计算F”(x)←F(x+r)+F’(x)。该计算对应于通过多项式F’(x)来掩蔽针对x的多项式F(x+r)的操作。应注意，在F(x+r)中，与r有关的信息仅出现在x的一次项中，因此与r有关的所有信息被F’(x)掩蔽。

[0375] 步骤1(续)：

[0376] 接下来，证明者算法P生成F’(z)和z的哈希值c_1。即，证明者算法P计算$c_1\leftarrow H_1(F'(z),z)$。此外，证明者算法P生成数w’的哈希值$c_2$。即，证明者算法P计算$c_2\leftarrow H_2(w')$。此外，证明者算法P生成多项式F”的哈希值$c_3$。即，证明者算法P计算$c_3\leftarrow H_3(F'')$。另外，所描述的$H_1(\cdots)$，$H_2(\cdots)$和$H_3(\cdots)$是哈希函数。此外，哈希值$(c_1,c_2,c_3)$是消息。

[0377] 在步骤1中生成的消息(c_1,c_2,c_3)被发送给验证者。

[0378] 步骤2：

[0379] 验证者算法V从三种验证模式中选择将被使用的验证模式。然后，验证者算法V向证明者发送指示所选验证模式的要求d∈{0,1,2}。

[0380] 步骤3：

[0381] 证明者算法P响应于从验证者接收的要求d来生成将被发送回验证者的信息σ。如果d＝0，则证明者算法P生成信息σ＝w。此外，如果d＝1，则证明者算法P生成信息σ＝(w’，z)。此外，如果d＝2，则证明者算法P生成信息σ＝(F”，z)。以这种方式生成的信息σ通过证明者算法P被发送给验证者。

[0382] 步骤4：

[0383] 验证者算法V利用从证明者接收的信息σ来执行下面的验证处理。

[0384] 如果d＝0，则验证者算法V计算(r’，w”)←G_1(σ)。此外，验证者算法V计算F”’←G_2(w”)。然后，验证者算法V验证等式c_2＝H_2(w”)是否成立。此外，验证者算法V验证等式c_3＝H_3(F(x+r’)+F”’(x))是否成立。验证者算法V在所有验证都已成功的情况中输出指示认证成功的值1，并且在在验证中出现了失败的情况中输出指示认证失败的值0。

[0385] 如果d＝1，则验证者算法V计算(w”，z’)←σ。此外，验证者算法V计算F”’←G_2(w”)。然后，验证者算法V验证等式c_1＝H_1(F”’(z’)，z’)是否成立。此外，验证者算法V验证等式c_2＝H_2(w”)是否成立。验证者算法V在所有验证都已成功的情况中输出指示认证成功的值1，并且在在验证中出现了失败的情况中输出指示认证失败的值0。

[0386] 如果d＝2，则验证者算法V计算(F”’，z’)←σ。然后，验证者算法V验证等式c_1＝H_1(F”’(z’)-y，z’)是否成立。此外，验证者算法V验证等式c_3＝H_3(F”’)是否成立。验证者算法V在所有验证都已成功的情况中输出指示认证成功的值1，并且在在验证中出现了失败的情况中输出指示认证失败的值0。

[0387] 在前面，已描述了在执行本方案时可假定的具体算法结构。

[0388] [2-7：高效算法]

[0389] n个变量的m个二次多项式的集合(f_1(x)，…，f_m(x))可被标识为如下的公式(12)。此外，x＝(x_1，…，x_n)是指示n个变量的向量。而且，A_1，…，A_m是n×n矩阵。而且，b_1，…，b_m是n×1向量。此外，c是m×1向量。

[0390] [表达式10]

[0391]

$$F(x)=\begin{pmatrix} f_1(x) \\ \vdots \\ f_m(x) \end{pmatrix}=\begin{pmatrix} x^T A_1 x+b_1^T x \\ \vdots \\ x^T A_m x+b_m^T x \end{pmatrix}+c \qquad \cdots(12)$$

[0392] 利用该表示，多项式的集合F可被标识为如下的公式(13)和公式(14)。可以从下面的公式(15)容易地确认该表示成立。

[0393] [表达式11]

[0394] $F(x_1+x_2)=F(x_1)+F(x_2)+F_b(x_1,x_2)-c$ …(13)

[0395] $$F_b(x_1,x_2)=\begin{pmatrix} x_2^T(A_1^T+A_1)x_1 \\ \vdots \\ x_2^T(A_m^T+A_m)x_1 \end{pmatrix} \quad \cdots(14)$$

[0396] $$f_l(x_1+x_2)=(x_1+x_2)^T A_l(x_1+x_2)+b_l^T(x_1+x_2)+c_l$$

[0397] $$=x_1^T A_l x_1+x_1^T A_l x_2+x_2^T A_l x_1+x_2^T A_l x_2+b_l^T x_1+b_l^T x_2+c_l$$

[0398] $$=f_l(x_1)+f_l(x_2)-c_l+x_1^T A_l x_2+x_2^T A_l x_1$$

[0399] $$=f_l(x_1)+f_l(x_2)-c_l+x_1^T(A_l^T)^T x_2+x_2^T A_l x_1$$

[0400] $$=f_l(x_1)+f_l(x_2)-c_l+(A_l^T x_1)^T x_2+x_2^T A_l x_1$$

[0401] $$=f_l(x_1)+f_l(x_2)-c_l+x_2^T(A_l^T x_1)+x_2^T A_l x_1$$

[0402] $$=f_l(x_1)+x_2^T(A_l^T+A_l)x_1+f_l(x_2)-c_l \quad \cdots(15)$$

[0403] 当将$F(x_1+x_2)$划分为三个部分时，即，取决于x_1的部分、取决于x_2的部分以及取决于x_1和x_2两者的部分，以这种方式，取决于x_1和x_2两者的部分$F_b(x_1,x_2)$将是相对于x_1和x_2的双线性映射。当利用该性质时，可以实现高效方案。

[0404] 例如，考虑本方案中的这样的修改：通过利用向量$t\in K^n$和$e\in K^m$来将多项式的集合$F'(x)$用于掩蔽与$F'(x)=F_b(x,t)+e$有关的多项式的集合$F(x+r)$。当这样的修改被执行时，与x有关的多项式的集合$F(x+r)$和$F'(x)$之和可被表示为如下的公式(16)。这里，当$t'=r+t$并且$e'=F(r)-c+e$时，与x有关的多项式$F''(x)$可以通过向量$t'\in K^n$和$e'\in K^m$来表示。出于这些原因，通过设置$F'(x)=F_b(x,t)+e$，F'和F''可由K^n上的向量以及K^m上的向量来表示，并且需要传输的数据的大小可被大幅减小。

[0405] [表达式12]

[0406] $F(x+r)+F'(x)$

[0407] $=F(x)+F(r)+F_b(x,r)-c+F_b(x,t)+e$

[0408] …(16)

[0409] $=F(x)+F_b(x,r+t)+F(r)-c+e$

[0410] 此外，通过上面的修改，与r有关的信息未从F''(或F')被泄露。例如，即使e'和t'(或者e和t)被给出，只要e和t(或者e'和t')不被知道，就无法知道与r有关的任何信息。因此，以上述方式进行了修改的本方案是保证了零知识的公钥认证方案。下面，将参考图9至11更详细地描述以上述方式被修改的根据本方案的高效算法。另外，密钥生成算法Gen的结构与上面所述的本方案的基本相同，因此将省略其详细说明。

[0411] 步骤1：

[0412] 首先，证明者算法P任意地选择一个数w。然后，证明者算法P通过将该数w应用于伪随机数生成器G_1来生成向量$r\in K^n$和数w'。即，证明者算法P计算$(r,w')\leftarrow G_1(w)$。接下来，证明者算法P通过将数w'应用于伪随机数生成器G_2来生成两个向量$t\in K^n$和$e\in K^m$。即，证明者

算法P计算(t,e)←G_2(w')。接下来,证明者算法P计算z←s-r。该计算对应于通过向量r来掩蔽私钥s的操作。此外,证明者算法P计算t'←r+t。接下来,证明者算法P计算e'←F(r)-c+e。

[0413] 步骤1(续):

[0414] 接下来,证明者算法P基于上面的公式(16)生成F_b(z,t),并且生成F_b(z,t)+e和z的哈希值c_1。即,证明者算法P计算c_1←H_1(F_b(z,t)+e,z)。此外,证明者算法P生成数w'的哈希值c_2。即,证明者算法P计算c_2←H_2(w')。此外,证明者算法P生成两个向量t'和e'的哈希值c_3。即,证明者算法P计算c_3←H_3(t',e')。另外,所描述的H_1(…),H_2(…)和H_3(…)是哈希函数。此外,哈希值(c_1,c_2,c_3)是消息。

[0415] 在步骤1中生成的消息(c_1,c_2,c_3)被发送给验证者。

[0416] 步骤2:

[0417] 验证者算法V从三种验证模式中选择将被使用的验证模式。然后,验证者算法V向证明者发送指示所选验证模式的要求d∈{0,1,2}。

[0418] 步骤3:

[0419] 证明者算法P响应于从验证者接收的要求d来生成将被发送回验证者的信息σ。如果d=0,则证明者算法P生成信息σ=w。此外,如果d=1,则证明者算法P生成信息σ=(w',z)。此外,如果d=2,则证明者算法P生成信息σ=(t',e',z)。以这种方式生成的信息σ通过证明者算法P被发送给验证者。

[0420] 步骤4:

[0421] 验证者算法V利用从证明者接收的信息σ来执行下面的验证处理。

[0422] 如果d=0,则验证者算法V计算(r',w")←G_1(σ)。此外,验证者算法V计算(t",e")←G_2(w")。然后,验证者算法V验证等式c_2=H_2(w")是否成立。此外,验证者算法V验证等式c_3=H_3(r'+t",F(r')-c+e")是否成立。验证者算法V在所有验证都已成功的情况中输出指示认证成功的值1,并且在在验证中出现了失败的情况中输出指示认证失败的值0。

[0423] 如果d=1,则验证者算法V计算(w",z')←σ。此外,验证者算法V计算(t",e")←G_2(w")。然后,验证者算法V验证等式c_1=H_1(F_b(z',t")+e",z')是否成立。此外,验证者算法V验证等式c_2=H_2(w")是否成立。验证者算法V在所有验证都已成功的情况中输出指示认证成功的值1,并且在在验证中出现了失败的情况中输出指示认证失败的值0。

[0424] 如果d=2,则验证者算法V计算(t‴,e‴,z')←σ。然后,验证者算法V验证等式c_1=H_1(F(z')+F_b(z',t‴)+e‴-y,z')是否成立。此外,验证者算法V验证等式c_3=H_3(t‴,e‴)是否成立。验证者算法V在所有验证都已成功的情况中输出指示认证成功的值1,并且在在验证中出现了失败的情况中输出指示认证失败的值0。

[0425] 在前面,已描述了本方案的高效算法。通过利用该高效算法,需要传输的数据的大小可被大幅减小。此外,由于将x+r代入多项式F并且关于x来重组多项式F的计算(F(x+r)的计算)在该交互式协议中成为不必要的,因此计算效率提高。

[0426] (高效算法的修改)

[0427] 现在,图9所示的本方案的高效算法可在维持计算效率的同时被修改 为如图10和11所示的算法。

[0428] (修改A:图10所示的算法)

[0429] 例如,计算e'←F(r)-c+e在图9所示的高效算法的步骤1中被执行,然而该计算可

被修改为如图10所示的计算e'←F(r)+e。然而，在执行这样的修改的情况中，将在步骤4中由验证者执行的验证的内容的一部分必须被修改。具体地，在d＝0的情况中将由验证者在步骤4执行的验证的内容中，对$c_3=H_3(r'+t'',F(r')-c+e'')$的验证必须由对$c_3=H_3(r'+t'',F(r')+e'')$的验证来取代。此外，在d＝1的情况中将由验证者在步骤4执行的验证的内容中，对$c_1=H_1(F(z')+F_b(z',t'')+e''-y,z')$的验证必须由对$c_1=H_1(F(z')+F_b(z',t'')-c+e''-y,z')$的验证来取代。

[0430] (修改B:图11所示的算法)

[0431] 此外，在图10所示的算法的步骤3中，在d＝0的情况中σ被设置为w，然而在d＝0的情况中被设置的该σ可以为任何信息，只要该信息是可从其恢复(r,t,e)的信息即可。例如，如图11所示，在d＝0的情况中将在步骤3被设置的σ的内容可以为(w',t')。然而，在执行这样的修改的情况中，将在步骤4中由验证者执行的验证的内容的一部分必须被修改。具体地，在d＝0的情况中将由验证者在步骤4执行的验证的内容中，对$c_3=H_3(r'+t'',F(r')+e'')$的验证必须由对$c_3=H_3(t',F(t'-t)+e'')$的验证来取代。

[0432] 如上所述，将在步骤1中由证明者执行的计算e'←F(r)-c+e可被改变为计算e'←F(r)+e(修改A)。此外，在d＝0的情况中将由证明者在步骤3设置的σ的内容可被改变为(w',t')等(修改B)。图11示出了应用了根据修改A的修改和根据修改B的修改两者的图9的高效算法，然而，也可以仅向图9的高效算法应用根据修改B的修改。即，修改A的修改内容和修改B的修改内容可以分离地被应用于图9的算法，或者这两者可以组合地被应用于图9的算法。

[0433] [2-8:多次多元联立方程式的形式]

[0434] 如上所述，本方案是采取求解多次多元联立方程式的难度作为安全性基础的方案。此外，本方案具有其特征:复杂的多次多元联立方程式可被 使用。在上面的说明中，未向多次多元联立方程式的形式施加任何特殊限制，然而优选地，例如使用体现了充分地保证其难度的元素加密技术的多次多元联立方程式。下面，将介绍可被应用于本方案的多次多元联立方程式的具体示例。

[0435] (2-8-1:与共同密钥块密码有关的形式)

[0436] 诸如AES、DES和KATAN之类的共同密钥块密码技术是得到良好地解析并且安全性和可靠性高的基本技术。这些共同密钥块密码可以通过以共同密钥块密码的密钥、明文和密文作为变量的多次多元联立方程式来表示。当值被赋予该多次多元联立方程式中的表示明文和密文的变量时，该多次多元联立方程式将成为仅以表示密钥的变量作为变量的方程式。

[0437] 求解表示该共同密钥块密码的这样的多次多元联立方程式对应于从明文和密文中恢复该共同密钥块密码的密钥。即，只要维持该共同密钥块密码的安全性，则求解表示该共同密钥块密码的多次多元联立方程式的难度就会得到保证。因此，如果表示共同密钥块密码的多次多元联立方程式被应用于本方案，则将实现具有与共同密钥块密码方案的安全性相同的安全性的公钥认证方案。

[0438] 然而，如果共同密钥块密码通过以密钥、明文和密文作为变量的多次多元联立方程式来表示，则多项式的次数将较高，因此，用于表示联立方程式的数据的大小将增加。因此，除了密钥、明文和密文以外，还将引入表示每轮(round)的内部状态的变量。通过引入该变量，可以使表示共同密钥块密码的多次多元联立方程式的次数变低。例如，适当的值被代

入表示明文和密文的变量中，并且与表示密钥和内部状态的变量有关的联立方程式被引入。通过采用这样的方法，变量的数目将增加，然而次数将降低，因此，联立方程式的表示将是紧凑的。

[0439] (2-8-2：与哈希函数有关的形式)

[0440] 类似地，与诸如SHA-1、SHA-256等哈希函数有关的多次多元联立方程式可被应用于本方案。这些哈希函数可以通过以消息(其是哈希函数的输入)和哈希值(其是输出)作为变量的多次多元联立方程式来表示。利用该多次多元联立方程式，当适当值被赋予表示哈希值的变量时，与表示 对应输入的变量有关的多次多元联立方程式被获得。

[0441] 求解这样的多次多元联立方程式对应于从哈希值恢复原始消息的值。即，只要维持哈希函数的安全性(单方向性)，求解表示该哈希函数的多次多元联立方程式的难度就会得到保证。因此，如果表示哈希函数的多次多元联立方程式被应用于本方案，则将实现基于哈希函数的安全性的公钥认证方案。

[0442] 然而，如果哈希函数通过以输入消息和哈希值作为变量的多次多元联立方程式来表示，则多项式的次数将较高，因此表示联立方程式的数据的大小将增加。因此，除了输入消息和哈希值以外，还将引入表示内部状态的变量。通过引入该变量，可以使表示哈希函数的多次多元联立方程式的次数变低。例如，适当的值被代入表示哈希值的变量中，并且与表示密钥和内部状态的变量有关的联立方程式被引入。通过采用这样的方法，变量的数目将增加，然而次数将降低，因此，联立方程式的表示将是紧凑的。

[0443] (2-8-3：与流密码有关的形式)

[0444] 类似地，与诸如Trivium之类的流密码有关的多次多元联立方程式可被应用于本方案。这些流密码可以通过与表示流密码的初始内部状态的变量以及表示作为输出的流的变量有关的多次多元联立方程式来表示。在此情况中，当适当值被赋予表示输出流的变量时，与表示对应初始内部状态的变量有关的多次多元联立方程式被获得。

[0445] 求解这样的多次多元联立方程式对应于从输出流的值中恢复表示原始的初始内部状态的变量。即，只要维持流密码的安全性，则求解表示该流密码的多次多元联立方程式的难度就会得到保证。因此，如果表示流密码的多次多元联立方程式被应用于本方案，则将实现基于流密码的安全性的公钥认证方案。

[0446] 然而，如果流密码通过以初始内部状态和输出流作为变量的多次多元联立方程式来表示，则多项式的次数将较高，因此表示联立方程式的数据的大小将增加。因此，除了初始内部状态和输出流以外，还将引入表示每轮的内部状态的变量。通过引入该变量，可以使表示流密码的多次多元联立方程式的次数变低。例如，适当的值被代入表示输出流的变量中，则与 表示初始内部状态和一轮的内部状态的变量有关的联立方程式被引入。通过采用这样的方法，变量的数目将增加，然而次数将降低，因此，联立方程式的表示将是紧凑的。

[0447] 在前面，已描述了本发明的第一实施例。

[0448] <3：第二实施例>

[0449] 接下来，将描述本发明的第二实施例。在前面，已描述了3次传递公钥认证方案。在本实施例中，将描述5次传递公钥认证方案(下面称为本方案)。本方案是用于通过使验证者的验证模式的数目为2q来确保公钥认证方案的合理性的方案。

[0450] 另外，在根据上述第一实施例的3次传递公钥认证方案中，每次交互式协议时的伪

造概率为2/3。然而，在本方案中，每次交互式协议时的伪造概率将为1/2+1/q，如后面将描述的。此外，q是将被使用的环的阶数。因此，如图27所示，如果环的阶数足够大，则本方案能够进一步减小每次交互式协议时的伪造概率，并且可以通过较小数目的交互式协议执行次数来充分地减小伪造概率。

[0451] 根据5次传递公钥认证方案的交互式协议可能看起来比根据3次传递公钥认证方案的交互式协议的效率低。然而，根据5次传递公钥认证方案，如果使环的阶数足够大，则每次交互式协议时的伪造概率将接近1/2，因此实现相同安全性等级所需的交互式协议的执行次数可以较少。

[0452] 例如，在3次传递公钥认证方案中，为了使伪造概率为$1/2^n$或更小，交互式协议必须被执行$n/(\log 3-1)=1.701n$次或更多次。另一方面，在5次传递公钥认证方案中，交互式协议必须被执行$n/(1-\log(1+1/q))$次或更多次。如图27所示，例如，当$q=24$时，对于5次传递公钥认证方案来说，实现相同安全性等级所需的通信量将比3次传递公钥认证方案少。

[0453] [3-1：公钥认证方案的算法]

[0454] 下面，将参考图12描述根据5次传递公钥认证方案(本方案)的算法结构。图12是用于描述根据本方案的算法的内容的说明图。

[0455] (密钥生成算法Gen)

[0456] 该密钥生成算法Gen生成在环K上定义的n个变量的m个多次多项式$f_1(x_1,\cdots,x_n),\cdots,f_m(x_1,\cdots,x_n)$以及向量$s=(s_1,\cdots,s_n)\in K^n$。接下来，该密钥生成算法Gen计算$y=(y_1,\cdots,y_m)\leftarrow(f_1(s),\cdots,f_m(s))$。然后，该密钥生成算法Gen设置$(f_1,\cdots,f_m,y)$作为公钥pk，并且设置s作为私钥。另外，在下面，n个变量的向量$(x_1,\cdots,x_n)$将被表达为x，并且n个变量的m个多次多项式$(f_1(x),\cdots,f_m(x))$将被表达为F(x)。

[0457] (证明者算法P、验证者算法V)

[0458] 接下来，将参考图12描述在交互式协议中通过证明者算法P和验证者算法V进行的处理。该交互式协议用于使得验证者证明“证明者知道满足$y=F(s)$的s”，而根本不会将关于s的信息泄露给验证者。另外，假设通过密钥生成算法Gen生成的公钥pk在证明者与验证者之间被共享。此外，假设通过密钥生成算法Gen生成的私钥sk由证明者秘密地管理。

[0459] 步骤1：

[0460] 首先，证明者算法P任意地选择一个数w。然后，证明者算法P通过将该数w应用于伪随机数生成器G来生成向量$r\in K^n$以及n个变量的多项式的集合$F'(x)=(f'_1(x),\cdots,f'_m(x))$。即，证明者算法P计算$(r,F')\leftarrow G(w)$。接下来，证明者算法P计算$z\leftarrow s-r$。该计算对应于通过向量r来掩蔽私钥s的操作。

[0461] 步骤1(续)：

[0462] 接下来，证明者算法P生成F'(z)和z的哈希值c_1。即，证明者算法P计算$c_1\leftarrow H_1(F'(z),z)$。此外，证明者算法P生成数w的哈希值$c_2$。即，证明者算法P计算$c_2\leftarrow H_2(w)$。另外，所描述的$H_1(\cdots)$和$H_2(\cdots)$是哈希函数。此外，哈希值$(c_1,c_2)$是消息。

[0463] 在步骤1中生成的消息(c_1,c_2)被发送给验证者。

[0464] 步骤2：

[0465] 验证者算法V从环K的q类元素中选择一个随机数α。然后，验证者算法V将所选随机数α发送给证明者。

[0466] 步骤3：

[0467] 证明者算法P计算F”(x)←αF(x+r)+F’(x)。该计算对应于通过多项式的集合F’(x)来掩蔽针对x的多项式的集合F(x+r)的操作。

[0468] 在步骤3中生成的多项式F”被发送给验证者。

[0469] 步骤4：

[0470] 验证者算法V从两种验证模式中选择将要使用的验证模式。然后，验证者算法V向证明者发送指示所选验证模式的要求d∈{0,1}。

[0471] 步骤5：

[0472] 证明者算法P响应于从验证者接收的要求d来生成将被发送回验证者的信息σ。如果d＝0，则证明者算法P生成信息σ＝w。此外，如果d＝1，则证明者算法P生成信息σ＝z。以这种方式生成的信息σ通过证明者算法P被发送给验证者。

[0473] 步骤6：

[0474] 验证者算法V利用从证明者接收的信息σ来执行下面的验证处理。

[0475] 如果d＝0，则验证者算法V计算(r’,F”’)←G(σ)。此外，验证者算法V验证等式c_2＝H_2(σ)是否成立。此外，验证者算法V验证等式F”(x)＝αF(x+r’)+F”’(x)是否成立。验证者算法V在所有验证都已成功的情况中输出指示认证成功的值1，并且在在验证中出现了失败的情况中输出指示认证失败的值0。

[0476] 如果d＝1，则验证者算法V计算z’←σ。此外，验证者算法V验证等式c_1＝H_1(F”’(z’)-αy,z’)是否成立。验证者算法V在验证已成功的情况中输出指示认证成功的值1，并且在在验证中出现了失败的情况中输出指示认证失败的值0。

[0477] (补充)

[0478] 此外，注意，在将上述步骤1和步骤3中生成的消息(c_1,c_2)以及多项式的集合F”发送给验证者时，与私钥sk有关的信息、与r有关的信息以及与z有关的信息根本不会被泄露给验证者。还注意，在将上述步骤5中生成的信息σ发送给验证者时，与z有关的信息在d＝0的情况中根本不会被泄露给验证者，并且与r有关的信息在d＝1的情况中根本不会被泄露给验证者。

[0479] (本方案中的合理性)

[0480] 本方案的合理性通过如下方式来得到保证：如果证明者算法P针对由验证者算法V选择的两种类型(α_1,α_2)以及集合(c_1,c_2)，对要求d＝0,1作出正确的响应，则满足以下公式(17)至(19)的F'''_1,F'''_2,F”’,r’和z’可从响应中被计算出。

[0481] [表达式13]

[0482] $F'''_1(x)=\alpha_1 F(x+r')+F'''(x)$ …(17)

[0483] $F'''_2(x)=\alpha_2 F(x+r')+F'''(x)$ …(18)

[0484] $F'''_1(z')-\alpha_1 y=F'''_2(z')-\alpha_2 y$ …(19)

[0485] 在这样的逻辑被保证的情况下，可以保证只要多次多元联立方程式不被求解出，就无法以高于1/2+1/q的概率执行伪造。即，为了正确地对验证者的所有要求d＝0,1作出响应，伪造者必须能够计算出满足上面公式(17)至(19)的F'''_1,F'''_2,F”’,r’和z’。换言之，伪造者必须能够计算出满足F(s)＝y的s。因此，只要多次多元联立方程式不被求解出，伪造者就不能够以高于1/2+1/q的概率成功地执行伪造。另外，通过将上述交互式协议执行足够多

的次数，伪造成功的概率可被减小到可忽略的水平。

[0486] （修改）

[0487] 上述密钥生成算法Gen计算$y \leftarrow F(s)$，并且将公钥设置为(F,y)。然而，该密钥生成算法Gen可被配置为获得$(y_1,\cdots,y_m) \leftarrow F(s)$并计算$(f_1^*(x),\cdots,f_m^*(x)) \leftarrow (f_1(x)-y_1,\cdots,f_m(x)-y_m)$，，并且将公钥设置为$(f_1^*,\cdots,f_m^*)$。通过该修改，使得交互式协议能够在取$y=0$的情况下在证明者与验证者之间被执行。

[0488] 此外，证明者算法P可以分开计算F”(z)的哈希值和z的哈希值，并且可以分开地将每个作为消息发送给验证者。

[0489] 此外，上述证明者算法P通过将伪随机数生成器G_1应用于数w来生成向量r和数w’。然后，证明者算法P通过将数w’应用于伪随机数生成器G_2.来生成n个变量的多项式F’(x)。然而，证明者算法P可以在最初计算出$w=(r,F')$，并且可以对G_1进行恒等映射。此外，在此情况中，数w不必被应用于G_1。另外，这也同样适用于G_2。

[0490] 在前面，已描述了根据本方案的基本算法结构。

[0491] [3-2:扩展算法]

[0492] 接下来，将参考图13描述作为本方案的扩展(下面称为扩展方案)的公钥认证方案的算法。图13是用于描述基于扩展方案的交互式协议的流程的说明图。该扩展方案是用于将在第三次传递中被发送的多项式的集合F”变换为一个哈希值c_3并将其发送给验证者的方案。通过这样的扩展，在交互式协议中其表示大小较大的多项式的集合F”被发送给验证者的概率可被减少为一半，并且将被传输的数据的大小可被减小。下面，将详细描述扩展方案的每个算法的内容。

[0493] （密钥生成算法Gen）

[0494] 该密钥生成算法Gen生成在环K上定义的n个变量的m个多次多项式$f_1(x_1,\cdots,x_n),\cdots,f_m(x_1,\cdots,x_n)$以及向量$s=(s_1,\cdots,s_n) \in K^n$。接下来，该密钥生成算法Gen计算$y=(y_1,\cdots,y_m) \leftarrow (f_1(s),\cdots,f_m(s))$。然后，该密钥生成算法Gen设置$(f_1,\cdots,f_m,y)$作为公钥pk，并且设置s作为私钥。另外，在下面，n个变量的向量$(x_1,\cdots,x_n)$将被表达为x，并且n个变量的m个多次多项式$(f_1(x),\cdots,f_m(x))$将被表达为F(x)。

[0495] （证明者算法P、验证者算法V）

[0496] 接下来，将参考图13描述在交互式协议中通过证明者算法P和验证者算法V进行的处理。该交互式协议用于使得验证者证明“证明者知道满足$y=F(s)$的s”，而根本不会将关于s的信息泄露给验证者。另外，假设通过密钥生成算法Gen生成的公钥pk在证明者与验证者之间被共享。此外，假设通过密钥生成算法Gen生成的私钥sk由证明者秘密地管理。

[0497] 步骤1：

[0498] 首先，证明者算法P任意地选择一个数w。然后，证明者算法P通过将该数w应用于伪随机数生成器G来生成向量$r \in K^n$和n个变量的多项式的集合F’(x)。即，证明者算法P计算$(r,F') \leftarrow G(w)$。接下来，证明者算法P计算$z \leftarrow s-r$。该计算对应于通过向量r来掩蔽私钥s的操作。

[0499] 步骤1(续)：

[0500] 接下来，证明者算法P生成F’(z)和z的哈希值c_1。即，证明者算法P计算$c_1 \leftarrow H_1(F'(z),z)$。此外，证明者算法P生成数w的哈希值$c_2$。即，证明者算法P计算$c_2 \leftarrow H_2(w)$。另外，所描

述的$H_1(\cdots)$和$H_2(\cdots)$是哈希函数。此外，哈希值(c_1,c_2)是消息。

[0501] 在步骤1生成的消息(c_1,c_2)被发送给验证者。

[0502] 步骤2：

[0503] 验证者算法V从环K的q类元素中选择一个随机数α。然后，验证者算法V将所选随机数α发送给证明者。

[0504] 步骤3：

[0505] 证明者算法P计算$F''(x)\leftarrow\alpha F(x+r)+F'(x)$。该计算对应于通过多项式的集合F'(x)来掩蔽针对x的多项式的集合F(x+r)的操作。此外，证明者算法P生成多项式的集合F"的哈希值c_3。即，证明者算法计算$c_3=H_3(F''(x))$。另外，所描述的$H_3(\cdots)$是哈希函数。此外，哈希值c_3是消息。

[0506] 在步骤3中生成的消息c_3被发送给验证者。

[0507] 步骤4：

[0508] 验证者算法V从两种验证模式中选择将要使用的验证模式。然后，验证者算法V向证明者发送指示所选验证模式的要求$d\in\{0,1\}$。

[0509] 步骤5：

[0510] 证明者算法P响应于从验证者接收的要求d来生成将被发送回验证者的信息σ。如果$d=0$，则证明者算法P生成信息$\sigma=w$。此外，如果$d=1$，则证明者算法P生成信息$\sigma=(z,F'')$。以这种方式生成的信息σ通过证明者算法P被发送给验证者。

[0511] 步骤6：

[0512] 验证者算法V利用从证明者接收的信息σ来执行下面的验证处理。

[0513] 如果$d=0$，则验证者算法V计算$(r',F''')\leftarrow G(\sigma)$。此外，验证者算法V验证等式$c_2=H_2(\sigma)$是否成立。此外，验证者算法V验证等式$c_3=H_3(\alpha F(x+r')+F'''(x))$是否成立。验证者算法V在所有验证都已成功的情况中输出指示认证成功的值1，并且在在验证中出现了失败的情况中输出指示认证失败的值0。

[0514] 如果$d=1$，则验证者算法V计算$(z',F''')\leftarrow\sigma$。此外，验证者算法V验证等式$c_1=H_1(F'''(z')-\alpha y,z')$是否成立。此外，验证者算法V验证等式$c_2=H_2(F'''(x))$是否成立。验证者算法V在验证已成功的情况中输出指示认证成功的值1，并且在在验证中出现了失败的情况中输出指示认证失败的值0。

[0515] 在前面，已描述了扩展方案的交互式协议中的每个算法的处理。通过这样的扩展，在交互式协议中其表示大小较大的多项式的集合F"被发送给验证者的概率可被减少为一半，并且将被传输的数据的大小可被减小。

[0516] [3-3：并行算法]

[0517] 现在，如已经描述的，当采用根据本方案或扩展方案的交互式协议时，伪造成功的概率可被减小至(1/2+1/q)或更小。因此，如果该交互式协议被执行两次，则伪造成功的概率可被减小至$((1/2+1/q))^2$或更小。以相同方式，如果该交互式协议被执行N次，伪造成功的概率变为$((1/2+1/q))^N$，并且如果N是足够大的数(例如，$N=80$)，则伪造成功的概率被减少到可忽略的水平。例如，将根据本方案的交互式协议并行地执行N次的算法在图14中示出。下面，将参考图14描述将交互式协议并行地执行N次的每个算法的内容。

[0518] (密钥生成算法Gen)

[0519] 密钥生成算法Gen生成在环K上定义的n个变量的m个多次多项式$f_1(x_1,\cdots,x_n),\cdots,f_m(x_1,\cdots,x_n)$以及向量$s=(s_1,\cdots,s_n)\in K^n$。接下来，该密钥生成算法Gen计算$y=(y_1,\cdots,y_m)\leftarrow(f_1(s),\cdots,f_m(s))$。然后，该密钥生成算法Gen设置$(f_1,\cdots,f_m,y)$作为公钥pk，并且设置s作为私钥。另外，在下面，n个变量的向量$(x_1,\cdots,x_n)$将被表达为x，并且n个变量的m个多次多项式$(f_1(x),\cdots,f_m(x))$将被表达为F(x)。

[0520] (证明者算法P、验证者算法V)

[0521] 接下来，将参考图14描述在交互式协议中通过证明者算法P和验证者算法V进行的处理。该交互式协议用于使得验证者证明“证明者知道满足$y=F(s)$的s”，而根本不会将关于s的信息泄露给验证者。另外，假设 通过密钥生成算法Gen生成的公钥pk在证明者与验证者之间被共享。此外，假设通过密钥生成算法Gen生成的私钥sk由证明者秘密地管理。

[0522] 步骤1：

[0523] 首先，证明者算法P针对$i=1$至N执行下面的处理(1)至处理(5)。(处理1)证明者算法P任意地选择一个数w_i。(处理2)证明者算法P通过将该数w_i应用于伪随机数生成器G来生成向量$r_i\in K^n$和多项式的集合$F'_i(x)$。即，证明者算法P计算$(r_i,F'_i)\leftarrow G(w_i)$。(处理3)证明者算法P计算$z_i\leftarrow s-r_i$。该计算对应于通过向量$r_i$来掩蔽私钥s的操作。

[0524] 步骤1(续)：

[0525] (处理4)证明者算法P生成$F'_i(z_i)$和z_i的哈希值$c_{1,i}$。即，证明者算法P计算$c_{1,i}\leftarrow H_1(F'_i(z_i),z_i)$。(处理5)证明者算法P生成数$w_i$的哈希值$c_{2,i}$。即，证明者算法P计算$c_{2,i}\leftarrow H_2(w_i)$。

[0526] 当针对$i=1$至N执行了上述(处理1)至(处理5)之后，在步骤1中生成的消息$(c_{1,i},c_{2,i})$($i=1$至N)被发送给验证者。

[0527] 步骤2：

[0528] 验证者算法V从环K的q类元素中选择N个随机数$\alpha_1,\cdots,\alpha_N$。然后，验证者算法V将所选随机数$\alpha_1,\cdots,\alpha_N$发送给证明者。

[0529] 步骤3：

[0530] 证明者算法P针对$i=1$至N来计算$F''_i(x)\leftarrow\alpha_iF(x+r_i)+F'_i(x)$。该计算对应于通过多项式的集合$F'_i(x)$来掩蔽针对x的多项式的集合$F(x+r_i)$的操作。然后，证明者算法P将多项式的集合$F''_1,\cdots,F''_N$发送给验证者。

[0531] 步骤4：

[0532] 验证者算法V针对$i=1$至N的每个，从两种验证模式中选择要使用的验证模式。然后，验证者算法V将指示所选验证模式的要求$d_i\in\{0,1\}$($i=1$至N)发送给证明者。

[0533] 步骤5：

[0534] 证明者算法P响应于从验证者接收的要求d_i来生成将被发送回验证者的信息σ_i。这里，证明者算法P针对$i=1$至N来执行下面的(处理1)或(处理2)。(处理1)如果$d_i=0$，则证明者算法P生成信息$\sigma_i=w_i$。(处理2)如果$d_i=1$，则证明者算法P生成信息$\sigma_i=z_i$。当上述(处理1)或(处理2)的决定和处理已被执行之后，信息σ_i($i=1$至N)通过证明者算法P被发送给验证者。

[0535] 步骤6：

[0536] 验证者算法V利用从证明者接收的信息σ_i($i=1$至N)来执行下面的验证处理。另

外，下面的处理针对i＝1至N被执行。

[0537] 如果$d_i=0$，则验证者算法V计算$(r'_i,F''_i)\leftarrow G(\sigma_i)$。然后，验证者算法V验证等式$c_{2,i}=H_2(\sigma_i)$是否成立。此外，验证者算法V验证等式$F''_i(x)=\alpha_i F(x+r'_i)+F''_i(x)$是否成立。验证者算法V在所有验证都已成功的情况中输出指示认证成功的值1，并且在在验证中出现了失败的情况中输出指示认证失败的值0。

[0538] 如果$d_i=1$，则验证者算法V计算$z'_i\leftarrow\sigma_i$。然后，验证者算法V验证等式$c_{1,i}=H_1(F''_i(z'_i),z'_i)$是否成立。此外，验证者算法V验证等式$c_{1,i}=H_1(F''_i(z'_i)-\alpha_i y,z_i)$是否成立。验证者算法V在所有验证都已成功的情况中输出指示认证成功的值1，并且在在验证中出现了失败的情况中输出指示认证失败的值0。

[0539] 在前面，已描述了并行地执行本方案的交互式协议的方法。如上所述，通过重复地执行本方案的交互式协议，伪造成功的概率可被减小到可忽略的水平。另外，可以同样地对扩展方案进行并行化。扩展方案的并行交互式协议的算法结构在图15中示出。

[0540] 另外，在上述步骤1之后，取代发送$(c_{1,1},c_{1,2},\cdots,c_{N,1},c_{N,2})$给验证者，它们的哈希值$H(c_{1,1},c_{1,2},\cdots,c_{N,1},c_{N,2})$可以总体地被发送。当采用这种配置时，在第一次传递中发送的消息将仅是一个哈希值，并且通信量可被大幅减少。此外，该哈希值以及未从发送自证明者的响应中被恢复为挑战的哈希值将与该响应一起从证明者被发送。根据这种配置，在并行重复N次的情况中，将被发送的信息的个数可被减少N-1个。

[0541] (根据扩展方案的并行算法)

[0542] 接下来，将参考图15描述根据扩展方案的并行算法的内容。另外，密钥生成算法Gen的结构与根据上面描述的本方案的并行算法的基本上相 同，因此将省略其详细说明。

[0543] 步骤1：

[0544] 首先，证明者算法P针对i＝1至N执行下面的处理(1)至处理(5)。(处理1)证明者算法P任意地选择一个数w_i。(处理2)证明者算法P通过将该数w_i应用于伪随机数生成器G来生成向量$r_i\in K^n$和多项式的集合$F'_i(x)$。即，证明者算法P计算$(r_i,F'_i)\leftarrow G(w_i)$。(处理3)证明者算法P计算$z_i\leftarrow s-r_i$。该计算对应于通过向量ri来掩蔽私钥s的操作。

[0545] 步骤1(续)：

[0546] (处理4)证明者算法P生成$F'_i(z_i)$和z_i的哈希值$c_{1,i}$。即，证明者算法P计算$c_{1,i}\leftarrow H_1(F'_i(z_i),z_i)$。(处理5)证明者算法P生成数$w'_i$的哈希值$c_{2,i}$。即，证明者算法P计算$c_{2,i}\leftarrow H_2(w'_i)$。

[0547] 当针对i＝1至N执行了上述(处理1)至(处理5)之后，在步骤1中生成的消息$(c_{1,i},c_{2,i})$(i＝1至N)被发送给验证者。

[0548] 步骤2：

[0549] 验证者算法V从环K的q类元素中选择N个随机数$\alpha_1,\cdots,\alpha_N$。然后，验证者算法V将所选随机数$\alpha_1,\cdots,\alpha_N$发送给证明者。

[0550] 步骤3：

[0551] 证明者算法P针对i＝1至N来计算$F''_i(x)\leftarrow\alpha_i F(x+r_i)+F'_i(x)$。该计算对应于通过多项式的集合$F'_i(x)$来掩蔽针对x的多项式的集合$F(x+r_i)$的操作。然后，证明者算法P生成多项式的集合$F''_1,\cdots,F''_N$的哈希值$c_3$。即，证明者算法P计算$c_3\leftarrow H_3(F''_1,\cdots,F''_N)$。另外，所描述的$H_3(\cdots)$是哈希函数。此外，哈希值$c_3$是消息。

[0552] 在步骤3中生成的哈希值c_3被发送给验证者。

[0553] 步骤4：

[0554] 验证者算法V针对i＝1至N的每个，从两种验证模式中选择要使用的验证模式。然后，验证者算法V将指示所选验证模式的要求$d_i \in \{0,1\}$(i＝1至N)发送给证明者。

[0555] 步骤5：

[0556] 证明者算法P响应于从验证者接收的要求d_i来生成将被发送回验证者 的信息σ_i。这里，证明者算法P针对i＝1至N来执行下面的(处理1)或(处理2)。(处理1)如果$d_i＝0$，则证明者算法P生成信息$\sigma_i＝w_i$。(处理2)如果$d_i＝1$，则证明者算法P生成信息$\sigma_i＝(z_i,F''_i)$。当上述(处理1)或(处理2)的决定和处理已被执行之后，信息σ_i(i＝1至N)通过证明者算法P被发送给验证者。

[0557] 步骤6：

[0558] 验证者算法V利用从证明者接收的信息σ_i(i＝1至N)来执行下面的验证处理。另外，下面的处理针对i＝1至N被执行。

[0559] 如果$d_i＝0$，则验证者算法V计算$(r'_i,F'''_i)\leftarrow G(\sigma_i)$。此外，验证者算法V计算$F''''_i \leftarrow \alpha_i F(x+r'_i)+F'''_i(x)$。然后，验证者算法V验证等式$c_{2,i}＝H_2(\sigma_i)$是否成立。然后，验证者算法V验证等式$c_3＝H_3(F''''_1,\cdots,F''''_N)$是否成立。验证者算法V在所有验证都已成功的情况中输出指示认证成功的值1，并且在在验证中出现了失败的情况中输出指示认证失败的值0。

[0560] 如果$d_i＝1$，则验证者算法V计算$(z'_i,F''''_i)\leftarrow\sigma_i$。然后，验证者算法V验证等式$c_{1,i}＝H_1(F''''_i(z'_i)-\alpha_i y,z'_i)$是否成立。此外，验证者算法V验证等式$c_3＝H_3(F''''_1,\cdots,F''''_N)$是否成立。验证者算法V在所有验证都已成功的情况中输出指示认证成功的值1，并且在在验证中出现了失败的情况中输出指示认证失败的值0。

[0561] 在前面，已描述了根据扩展方案的并行算法的内容。

[0562] (参数设置的合适方法)

[0563] 现在，与根据上述第一实施例的交互式协议一样，根据本实施例的交互式协议保证了针对被动攻击的安全性。然而，在采用并行地重复执行该交互式协议的上述方法的情况中，为了保证针对主动攻击的安全性得到绝对保证，必须满足如下所述的条件。

[0564] 上述交互式协议是证明者利用一对密钥(公钥y、私钥s)向验证者证明“对于y，该证明者知道满足$y＝F(s)$的s”的方案。因此，如果通过验证而被接受的交互被执行，则不能否认这样的可能性，即“证明者在交互时使用了该s”的信息将被验证者所知。此外，对于上面的F，抗冲突能力得不到保证。因此，对于在无条件的情形下并行地重复执行上述交互式协议的情况，难以证明针对主动攻击的安全性将得到绝对保证。

[0565] 因此，本申请的发明人已调研出了一种方法，该方法即使在通过验证而被接受的交互被执行时也可防止“证明者在交互时使用了该s”的信息被验证者所知。于是，本申请的发明人已想出了即使在并行地重复执行上述交互式协议的情况中也能够保证针对主动攻击的安全性的方法。该方法用于将用作公钥的n个变量的多次多项式$f_1,\cdots,f_m$的数目m设置为充分小于变量的数目n的值。例如，m和n被设置为使得$2^{m-n}<<1$(例如，当n＝160并且m＝80时，$2^{-80}<<1$)。

[0566] 在采取求解多次多元方程式的难度作为安全性基础的上述方案中，即使私钥s_1和与该私钥相对应的公钥pk被给出，也难以生成与该公钥pk相对应的另一私钥s_2。因此，如果

保证了对于公钥pk存在两个以上的私钥s，则即使当通过验证而被接受的交换被执行时，也可以防止“证明者在交互时使用了该s”的信息被验证者所知。即，如果该保证成立，则即使在并行地重复执行交互式协议的情况中，针对主动攻击的安全性也可以得到保证。

[0567] 参考图28，考虑由n个变量的m个多次多项式形成的函数$F:K^n \rightarrow K^m$（其中，$n>m$），不具有第二前像的定义域中的元素数目最大仅为$|K|^m-1$。因此，如果使$|K|^{m-n}$充分小，则不具有第二前像的定义域中的元素被选择的可能性可以被降低到可忽略的水平。即，如果n个变量的多次多项式$f_1,\cdots,f_m$的数目m被设置为充分小于变量的数目n的值，则可以保证对于公钥pk存在两个以上私钥s。结果，同样在通过验证而被接受的交互被执行的情况中，可以防止“证明者在交互时使用了该s”的信息被验证者所知，并且即使在并行地重复执行交互式协议的情况中，针对主动攻击的安全性也将得到保证。

[0568] 如上所述，通过采用将n个变量的多次多项式$f_1,\cdots,f_m$的数目m设置为小于变量的数目n的值（$n>m$；优选地，$2^{m-n}<<1$）的方法，可以提高并行地重复执行交互式协议的情况的安全性。

[0569] [3-4：非交互式算法]

[0570] 在前面，已描述了5次传递公钥认证方案。然而，在根据本方案中，从验证者发送给证明者的信息实际上仅仅是随机数，因此可以修改为1次传递公钥认证方案（下面称为非交互式方案）。另外，根据非交互式方案的每个算法的内容在图16中示出。下面，将参考图16描述根据非交互式方案的每个算法的内容。

[0571] （密钥生成算法Gen）

[0572] 密钥生成算法Gen生成在环K上定义的n个变量的m个多次多项式$f_1(x_1,\cdots,x_n),\cdots,f_m(x_1,\cdots,x_n)$以及向量$s=(s_1,\cdots,s_n)\in K^n$。接下来，该密钥生成算法Gen计算$y=(y_1,\cdots,y_m)\leftarrow(f_1(s),\cdots,f_m(s))$。然后，该密钥生成算法Gen设置$(f_1,\cdots,f_m,y)$作为公钥pk，并且设置s作为私钥。另外，在下面，n个变量的向量$(x_1,\cdots,x_n)$将被表达为x，并且n个变量的m个多次多项式$(f_1(x),\cdots,f_m(x))$将被表达为F(x)。

[0573] （证明者算法P、验证者算法V）

[0574] 接下来，将参考图16描述在交互式协议中通过证明者算法P和验证者算法V进行的处理。该交互式协议用于使得验证者证明“证明者知道满足$y=F(s)$的s”，而根本不会将关于s的信息泄露给验证者。另外，假设通过密钥生成算法Gen生成的公钥pk在证明者与验证者之间被共享。此外，假设通过密钥生成算法Gen生成的私钥sk由证明者秘密地管理。

[0575] 步骤1：

[0576] 首先，证明者算法P针对$i=1$至N执行下面的处理(1)至处理(5)。(处理1)证明者算法P任意地选择一个数w_i。(处理2)证明者算法P通过将该数w_i应用于伪随机数生成器G来生成向量$r_i\in K^n$和多项式的集合$F'_i(x)$。即，证明者算法P计算$(r_i,F'_i)\leftarrow G(w_i)$。(处理3)证明者算法P计算$z_i\leftarrow s-r_i$。该计算对应于通过向量$r_i$来掩蔽私钥s的操作。

[0577] 步骤1(续)：

[0578] (处理4)证明者算法P生成$F_i(z_i)$和z_i的哈希值$c_{1,i}$。即，证明者算法P计算$c_{1,i}\leftarrow H_1(F_i(z_i),z_i)$。(处理5)证明者算法P生成数$w_i$的哈希值$c_{2,i}$。即，证明者算法P计算$c_{2,i}\leftarrow H_2(w_i)$。

[0579] 步骤1(续)：

[0580] 接下来，证明者算法P选择随机数R_A和R_B。然后，证明者算法P通过将随机数R_A以及通过上面的(处理4)和(处理5)计算出的哈希值$c_{1,i}$和$c_{2,i}$应用于哈希函数H_A来生成哈希值$\alpha_1,\cdots,\alpha_N$。即，证明者算法P计算$(\alpha_1,\cdots,\alpha_N)\leftarrow H_A(R_A,c_{1,1},\cdots,c_{2,N})$。

[0581] 步骤1(续)：

[0582] 接下来，证明者算法P针对i＝1至N执行下面的(处理1)和(处理2)。(处理1)证明者算法P计算$F''_i(x)\leftarrow\alpha_i F(x+r_i)+F'_i(x)$。该计算对应于通过多项式的集合$F'_i(x)$来掩蔽针对x的多项式的集合$F(x+r_i)$的操作。(处理2)证明者算法P针对i＝1至N，通过将随机数R_A和R_B、哈希值($c_{1,1}$和$c_{2,i}$)、α_i以及F''_i应用于哈希函数H_B来生成$d=(d_1,\cdots,d_N)$。即，证明者算法P计算$(d_1,\cdots,d_N)\leftarrow H_B(R_A,R_B,c_{1,1},\cdots,c_{2,N},\alpha_1,\cdots,\alpha_N,F''_1,\cdots,F''_N)$。

[0583] 步骤1(续)：

[0584] 接下来，证明者算法P根据所生成的d_i来生成将被发送给验证者的信息σ_i。这里，证明者算法P针对i＝1至N来执行下面的(处理1)或(处理2)。(处理1)如果$d_i=0$，则证明者算法P生成信息$\sigma_i=w_i$。(处理2)如果$d_i=1$，则证明者算法P生成信息$\sigma_i=z_i$。当上述(处理1)或(处理2)的决定和处理已被执行之后，$R_A,R_B,c_{1,i},c_{2,i},\sigma_i$(i＝1至N)通过证明者算法P被发送给验证者。

[0585] 步骤2：

[0586] 验证者算法V首先通过将接收自证明者的$R_A,c_{1,i}$和$c_{2,i}$应用于哈希函数H_A来生成α_i。即，验证者算法V计算$(\alpha_1,\cdots,\alpha_N)\leftarrow H_A(R_A,c_{1,1},\cdots,c_{2,N})$。接下来，验证者算法V计算$d=(d_1,\cdots,d_N)\leftarrow H_B(R_A,R_B,c_{1,1},\cdots,c_{2,N},\alpha_1,\cdots,\alpha_N,F''_1,\cdots,F''_N)$。然后，验证者算法V通过利用信息$\sigma_i$(i＝1至N)来执行下面的验证处理。另外，下面的处理针对i＝1至N被执行。

[0587] 如果$d_i=0$，则验证者算法V计算$(r'_i,F'''_i)\leftarrow G(\sigma_i)$。然后，验证者算法V验证等式$c_{2,i}=H_2(\sigma_i)$是否成立。此外，验证者算法V验证等式$F''_i(x)=\alpha_i F(x+r'_i)+F'''_i(x)$是否成立。验证者算法V在所有验证都已成功的情况中输出指示认证成功的值1，并且在在验证中出现了失败的情况中输出指示认 证失败的值0。

[0588] 如果$d_i=1$，则验证者算法V计算$z'_i\leftarrow\sigma_i$。然后，验证者算法V验证等式$c_{1,i}=H_1(F'''_i(z'_i)-\alpha_i y,z'_i)$是否成立。验证者算法V在该验证成功的情况中输出指示认证成功的值1，并且在在该验证出现了失败的情况中输出指示认证失败的值0。

[0589] 在前面，已描述了根据非交互式方案的每个算法的内容。另外，如果理想的哈希函数H_A和H_B被假定，则哈希值α_i和d_i将随机地表现，因此，方便于证明者的哈希值α_i和d_i将不会出现。因此，即使在如上所述的修改被执行时，也将确保足够的安全性。另外，这样的修改同样可以应用于扩展方案等。

[0590] [3-5：修改为数字签名方案]

[0591] 这里，将描述将本方案修改为数字签名方案的方法。另外，出于简化考虑，这里将描述将上述非交互式方案修改为数字签名方案的方法。可以明白，当上述非交互式方案的证明者和验证者对应于数字签名方案的签名者和验证者时，对于数字签名方案的模型来说存在的类似性在于只有证明者可以取信验证者。考虑到该概念，将详细描述基于非交互式方案的数字签名方案的算法结构。

[0592] (密钥生成算法Gen)

[0593] 该密钥生成算法Gen生成在环K上定义的n个变量的m个多次多项式$f_1(x_1,\cdots,$

$x_n),\cdots,f_m(x_1,\cdots,x_n)$以及向量$s=(s_1,\cdots,s_n)\in K^n$。接下来，该密钥生成算法Gen计算$y=(y_1,\cdots,y_m)\leftarrow(f_1(s),\cdots,f_m(s))$。然后，该密钥生成算法Gen设置$(f_1,\cdots,f_m,y)$作为公钥pk，并且设置s作为私钥。另外，在下面，n个变量的向量$(x_1,\cdots,x_n)$将被表达为x，并且n个变量的m个多次多项式$(f_1(x),\cdots,f_m(x))$将被表达为F(x)。

[0594] (签名生成算法Sig)

[0595] 该签名生成算法Sig针对i＝1至N来执行下面的(处理1)至(处理11)。另外，假设签名密钥sk＝s和文档M被输入签名生成算法Sig中。

[0596] (处理1)签名生成算法Sig任意地选择一个数w_i。(处理2)签名 生成算法Sig通过将该数w_i应用于伪随机数生成器G来生成向量$r_i\in K^n$和n个变量的多项式的集合$F'_i(x)=(f_{1,1}'(x),\cdots,f_{m,i}'(x))$。即，签名生成算法Sig计算$(r_i,F'_i)\leftarrow G(w_i)$。(处理2)签名生成算法Sig计算$z_i\leftarrow s_i-r_i$。该计算对应于通过向量$r_i$来掩蔽私钥s的操作。

[0597] (处理3)签名生成算法Sig生成$F''_i(z_i)$和z_i的哈希值$c_{1,i}$。即，签名生成算法Sig计算$c_{1,i}\leftarrow H_1(F''_i(z_i),z_i)$。(处理4)签名生成算法Sig生成数$w'_i$的哈希值$c_{2,i}$。即，签名生成算法Sig计算$c_{2,i}\leftarrow H_2(w'_i)$。另外，所描述的$H_1(\cdots)$和$H_2(\cdots)$是哈希函数。

[0598] (处理4)签名生成算法Sig任意地选择随机数R_A。(处理5)签名生成算法Sig计算$\alpha=(\alpha_1,\cdots,\alpha_N)\leftarrow H_A(R_A,M,c_{1,1},\cdots,c_{2,N})$。(处理6)签名生成算法Sig对于i＝1至N，计算$F''_i(x)=\alpha_iF(x+r'_i)+F'_i(x)$。(处理7)签名生成算法Sig任意地选择随机数$R_B$。(处理8)签名生成算法Sig计算$d=(d_1,\cdots,d_N)\leftarrow H_B(R_B,M,c_{1,1},\cdots,c_{2,N},\alpha,F''_1,\cdots,F''_N)$。

[0599] 接下来，该签名生成算法Sig针对i＝1至N，根据d_i来执行接下来的(处理9)或(处理10)。(处理9)如果$d_i=0$，则签名生成算法Sig计算$\sigma_i\leftarrow w_i$。(处理10)如果$d_i=1$，则签名生成算法Sig计算$\sigma_i\leftarrow z_i$。

[0600] (处理11)在针对i＝1至N执行了上面的(处理9)或(处理10)的决定和处理之后，签名生成算法Sig输出数字签名$\sigma=(R_A,R_B,c_{1,1},\cdots,c_{2,N},\alpha_1,\cdots,\alpha_N,F''_1,\cdots,F''_N,\sigma_1,\cdots,\sigma_N)$。

[0601] (签名验证算法Ver)

[0602] 如果对于i＝1至N下面的验证都通过了，则签名验证算法Ver接受该数字签名σ，并且如果即使一个验证未通过，则签名验证算法Ver也否决该数字签名σ。另外，假设数字签名σ和文档M被输入签名验证算法Ver中。首先，该签名验证算法Ver计算$\alpha=(\alpha_1,\cdots,\alpha_N)\leftarrow H_A(R_A,M,c_{1,1},\cdots,c_{3,N})$。接下来，该签名验证算法Ver计算$d=(d_1,\cdots,d_N)\leftarrow H(R,M,c_{1,1},\cdots,c_{3,N},\alpha,F''_1,\cdots,F''_N)$。然后，该签名验证算法Ver针对i＝1至N来执行下面的(验证1)和(验证2)。

[0603] (验证1)如果$d_i=0$，则签名验证算法Ver计算$(r'_i,F''_i)\leftarrow G(\sigma_i)$。然后，签名验证算法Ver验证等式$c_{2,i}=H_2(\sigma_i)$是否成立。此外，签名验证算 法Ver验证等式$F''_i(x)=\alpha_iF(x+r'_i)+F'_i(x_1,\cdots,x_n)$是否成立。签名验证算法Ver在所有验证都已成功的情况中输出指示接受该数字签名σ的值1，并且在在验证中出现了失败的情况中输出指示否决该数字签名σ的值0。

[0604] (验证2)如果$d_i=1$，则签名验证算法Ver计算$(z'_i,F'''_i)\leftarrow\sigma_i$。然后，签名验证算法Ver验证等式$c_{1,i}=H_1(F'''_i(z'_i),z'_i)$是否成立。签名验证算法Ver在该验证成功的情况中输出指示接受该数字签名σ的值1，并且在在验证中出现了失败的情况中输出指示否决该数字签名σ的值0。

[0605] 在前面，已描述了基于本方案的数字签名方案的每个算法结构。基于上述扩展方

案的数字签名方案可以同样地被构成。

[0606] [3-6:具体示例]

[0607] 接下来,将参考图17描述在执行本方案时可假定的具体算法结构。图17是用于描述本方案的具体示例的说明图。

[0608] (密钥生成算法Gen)

[0609] 该密钥生成算法Gen生成在环K上定义的n个变量的m个二次多项式$f_1(x_1,\cdots,x_n),\cdots,f_m(x_1,\cdots,x_n)$以及向量$s=(s_1,\cdots,s_n)\in K^n$。接下来,该密钥生成算法Gen计算$y=(y_1,\cdots,y_m)\leftarrow(f_1(s),\cdots,f_m(s))$。然后,该密钥生成算法Gen设置$(f_1,\cdots,f_m,y)$作为公钥pk,并且设置s作为私钥。另外,在下面,n个变量的向量$(x_1,\cdots,x_n)$将被表达为x,并且n个变量的m个二次多项式$(f_1(x),\cdots,f_m(x))$将被表达为F(x)。

[0610] (证明者算法P、验证者算法V)

[0611] 接下来,将参考图17描述在交互式协议中通过证明者算法P和验证者算法V进行的处理。该交互式协议用于使得验证者证明"证明者知道满足y=F(s)的s",而根本不会将关于s的信息泄露给验证者。另外,假设通过密钥生成算法Gen生成的公钥pk在证明者与验证者之间被共享。此外,假设通过密钥生成算法Gen生成的私钥sk由证明者秘密地管理。

[0612] 步骤1:

[0613] 首先,证明者算法P任意地选择一个数w。然后,证明者算法P通过将该数w应用于伪随机数生成器G来生成向量$r\in K^n$和n个变量的多项式 的集合$F'(x)=(f'_1(x),\cdots,f'_m(x))$。即,证明者算法P计算$(r,F')\leftarrow G(w)$。接下来,证明者算法P计算$z\leftarrow s-r$。该计算对应于通过向量r来掩蔽私钥s的操作。另外,多项式被表示为如下的公式(20)。

[0614] $$f'_i(x)=\sum_j b'_{i,j}x_j+c'_i \qquad \cdots(20)$$

[0615] 步骤1(续):

[0616] 接下来,证明者算法P生成F'(z)和z的哈希值c_1。即,证明者算法P计算$c_1\leftarrow H_1(F'(z),z)$。此外,证明者算法P生成数w的哈希值$c_2$。即,证明者算法P计算$c_2\leftarrow H_2(w)$。另外,所描述的$H_1(\cdots)$和$H_2(\cdots)$是哈希函数。此外,哈希值$(c_1,c_2)$是消息。

[0617] 在步骤1中生成的消息(c_1,c_2)被发送给验证者。

[0618] 步骤2:

[0619] 验证者算法V从环K的q类元素中选择一个随机数α。然后,验证者算法V将所选随机数α发送给证明者。

[0620] 步骤3:

[0621] 证明者算法P计算$F''(x)\leftarrow\alpha F(x+r)+F'(x)$。该计算对应于通过多项式的集合F'(x)来掩蔽针对x的多项式的集合F(x+r)的操作。

[0622] 在步骤3中生成的多项式F"被发送给验证者。

[0623] 步骤4:

[0624] 验证者算法V从两种验证模式中选择将要使用的验证模式。然后,验证者算法V向证明者发送指示所选验证模式的要求$d\in\{0,1\}$。

[0625] 步骤5:

[0626] 证明者算法P响应于从验证者接收的要求d来生成将被发送回验证者的信息σ。如

果d＝0，则证明者算法P生成信息σ＝w。此外，如果d＝1，则证明者算法P生成信息σ＝z。以这种方式生成的信息σ通过证明者算法P被发送给验证者。

[0627] 步骤6：

[0628] 验证者算法V利用从证明者接收的信息σ来执行下面的验证处理。

[0629] 如果d＝0，则验证者算法V计算$(r',F'')\leftarrow G(\sigma)$。此外，验证者算法V验证等式$c_2=H_2(\sigma)$是否成立。此外，验证者算法V验证等式$F''(x)=\alpha F(x+r')+F''(x)$是否成立。验证者算法V在所有验证都已成功的情况中输出指示认证成功的值1，并且在在验证中出现了失败的情况中输出指示认证失败的值0。

[0630] 如果d＝1，则验证者算法V计算$z'\leftarrow\sigma$。此外，验证者算法V验证等式$c_1=H_1(F''(z')-\alpha y,z')$是否成立。验证者算法V在该验证成功的情况中输出指示认证成功的值1，并且在在验证中出现了失败的情况中输出指示认证失败的值0。

[0631] 前面，已描述了在执行本方案时可假定的具体算法结构。

[0632] [3-7：高效算法]

[0633] 这里，与上述第一实施例一样，将考虑高效算法。例如，将考虑这样的方法：通过利用两个向量$t\in K^n$和$e\in K^m$来将用于掩蔽针对x的多项式的集合F(x+r)的多项式集合F'(x)表示为$F'(x)=F_b(x,t)+e$。当使用该方法时，针对x的多项式的集合F(x+r)被表示为如下的公式(21)。

[0634] [表达式15]

[0635] $\alpha F(x+r)+F'(x)$

[0636] $=\alpha F(x)+\alpha F(r)+\alpha F_b(x,r)-\alpha c+F_b(x,t)+e$

[0637] …(21)

[0638] $=\alpha F(x)+F_b(x,\alpha r+t)+\alpha(F(r)-c)+e$

[0639] 因此，当$t'=\alpha r+t$并且$e'=\alpha(F(r)-c)+e$时，掩蔽之后的与x有关的多项式的集合F''(x)也可以通过两个向量$t'\in K^n$和$e\in K^m$来表示。出于这些原因，当F'被设置为$F'(x)=F_b(x,t)+e$时，F'和F''将由K^n上的向量以及K^m上的向量来表示，并且需要传输的数据的大小可以大幅减小。

[0640] 此外，通过上面的修改，与r有关的信息未从F''(或F')被泄露。例如，即使e'和t'(或者e和t)被给出，只要e和t(或者e'和t')不被知道，就无法知道与r有关的任何信息。因此，以上述方式进行了修改的本方案是保证了零知识的公钥认证方案。下面，将参考图18至25更详细地描述以上述方式被修改的根据本方案的高效算法。另外，密钥生成算法Gen的结构与上面所述的本方案的基本相同，因此将省略其详细说明。

[0641] 步骤1：

[0642] 首先，证明者算法P任意地选择一个数w。然后，证明者算法P通过将该数w应用于伪随机数生成器G来生成向量$r\in K^n$，$t\in K^n$和$e\in K^m$。即，证明者算法P计算$(r,t,e)\leftarrow G(w)$。接下来，证明者算法P计算$z\leftarrow s-r$。该计算对应于通过向量r来掩蔽私钥s的操作。

[0643] 步骤1(续)：

[0644] 接下来，证明者算法P生成$F_b(z,t)+e$和z的哈希值c_1。即，证明者算法P计算$c_1\leftarrow H_1(F_b(z,t)+e,z)$。此外，证明者算法P生成数w的哈希值$c_2$。即，证明者算法P计算$c_2\leftarrow H_2(w)$。另外，所描述的$H_1(\cdots)$和$H_2(\cdots)$是哈希函数。此外，哈希值$(c_1,c_2)$是消息。

[0645] 在步骤1中生成的消息(c_1,c_2)被发送给验证者。

[0646] 步骤2：

[0647] 验证者算法V从环K的q类元素中选择一个随机数α。然后，验证者算法V将所选随机数α发送给证明者。

[0648] 步骤3：

[0649] 证明者算法P计算t'←αr+t。此外，证明者算法P计算e'←α(F(r)-c)+e。然后，证明者算法P将t'和e'发送给验证者。

[0650] 步骤4：

[0651] 验证者算法V从两种验证模式中选择将要使用的验证模式。然后，验证者算法V向证明者发送指示所选验证模式的要求d∈{0,1}。

[0652] 步骤5：

[0653] 证明者算法P响应于从验证者接收的要求d来生成将被发送回验证者的信息σ。如果d＝0，则证明者算法P生成信息σ＝w。此外，如果d＝1，则证明者算法P生成信息σ＝z。以这种方式生成的信息σ通过证明者算法P被发送给验证者。

[0654] 步骤6：

[0655] 验证者算法V利用从证明者接收的信息σ来执行下面的验证处理。

[0656] 如果d＝0，则验证者算法V计算(r',t",e")←G(σ)。然后，验证者算法V验证等式c_2＝H_2(σ)是否成立。此外，验证者算法V验证等式t'＝αr'+t"是否成立。此外，验证者算法V验证等式e'＝α(F(r')-c)+e"是否成立。验 证者算法V在所有验证都已成功的情况中输出指示认证成功的值1，并且在在验证中出现了失败的情况中输出指示认证失败的值0。

[0657] 如果d＝1，则验证者算法V计算z'←σ。然后，验证者算法V验证等式c_1＝H_1(αF(z')-y+F_b(z',t')+e',z')是否成立。验证者算法V在该验证成功的情况中输出指示认证成功的值1，并且在在验证中出现了失败的情况中输出指示认证失败的值0。

[0658] 在前面，已描述了本方案的高效算法的结构。通过利用该高效算法，需要用于传输的数据的大小可被大幅减小。此外，由于将x+r代入多项式F并且关于x来重组多项式F的计算(F(x+r)的计算)在该交互式协议中成为不必要的，因此计算效率提高。

[0659] (高效算法的修改)

[0660] 现在，图18所示的本方案的高效算法可在维持计算效率的同时被修改为如图19至25所示的算法。

[0661] (修改A:图19所示的算法)

[0662] 例如，计算e'←α(F(r)-c)+e在图18所示的高效算法的步骤3中被执行，然而该计算可被修改为如图19所示的计算e'←αF(r)+e。然而，在执行这样的修改的情况中，将在步骤6中由验证者执行的验证的内容的一部分必须被修改。具体地，在d＝0的情况中将由验证者在步骤6执行的验证的内容中，对e'＝α(F(r')-c)+e"的验证必须由对e'＝αF(r')+e"的验证来取代。此外，在d＝2的情况中将由验证者在步骤6执行的验证的内容中，对c_1＝H_1(α(F(z')-y)+F_b(z',t"')+e"',z')的验证必须由对c_1＝H_1(α(F(z')-c-y)+F_b(z',t"')+e"',z')的验证来取代。

[0663] (修改B:图20所示的算法)

[0664] 此外，在图19所示的算法的步骤5中，在d＝0的情况中σ被设置为w，然而在d＝0的

情况中被设置的该σ可以为任何信息，只要该信息是当与(t',e")一起使用时可从其恢复(r,t,e)的信息即可。例如，如图20所示，在d＝0的情况中将在步骤5被设置的σ的内容可以为r。然而，在执行这样的修改的情况中，图19所示的算法的步骤1中的计算$c_2 \leftarrow H_2(w)$必须被修改为$c_2 \leftarrow H_2(r,t,e)$。此外，在d＝0的情况中将由验证者在步骤6执行的 验证的内容中必须由对$c_2 = H_2(r,t'-\alpha r,e'-\alpha F(r))$的验证取代。

[0665] (修改C:图21所示的算法)

[0666] 例如，计算$t' \leftarrow \alpha r+t$在图20所示的算法的步骤3中被执行，然而该计算可被修改为如图21所示的计算$t' \leftarrow \alpha(r+t)$。然而，在执行这样的修改的情况中，在d＝0的情况下在图20所示的算法的步骤6中由验证者执行的验证的内容必须由对$c_2 = H_2(r,\alpha^{-1}t'-r,e'-\alpha F(r))$的验证来取代。

[0667] (修改D:图22所示的算法)

[0668] 例如，计算$e' \leftarrow \alpha F(r)+e$在图20所示的算法的步骤3中被执行，然而该计算可被修改为如图22所示的计算$e' \leftarrow \alpha(F(r)+e)$。然而，在执行这样的修改的情况中，在d＝0的情况下在图20所示的算法的步骤6中由验证者执行的验证的内容必须由对$c_2 = H_2(r,t'-\alpha r,e'-\alpha^{-1}e'-F(r))$的验证来取代。

[0669] (修改E:图23所示的算法)

[0670] 此外，在图20所示的算法的步骤5中，在d＝0的情况中σ被设置为r，然而在d＝0的情况中被设置的该σ可以为任何信息，只要该信息是当与(t',e")一起使用时可从其恢复(r,t,e)的信息即可。例如，如图23所示，在d＝0的情况中将在步骤5被设置的σ的内容可以为t。然而，在执行这样的修改的情况中，必须从步骤2的$\alpha \in_R K \setminus \{0\}$中来选择α，并且此外，在d＝0的情况中将由验证者在步骤6执行的验证的内容必须由对$c_2 = H_2(\alpha^{-1}(t'-t),t,e'-\alpha F(\alpha^{-1}(t'-t)))$的验证来取代。

[0671] (修改F:图24所示的算法)

[0672] 例如，计算$t' \leftarrow \alpha r+t$在图23所示的算法的步骤3中被执行，然而该计算可被修改为如图24所示的计算$t' \leftarrow \alpha(r+t)$。然而，在执行这样的修改的情况中，在d＝0的情况下在图23所示的算法的步骤6中由验证者执行的验证的内容必须由对$c_2 = H_2(\alpha^{-1}t'-t,t,e'-\alpha F(\alpha^{-1}t'-t))$的验证来取代。

[0673] (修改G:图25所示的算法)

[0674] 例如，计算$e' \leftarrow \alpha F(r)+e$在图23所示的算法的步骤3中被执行，然而该计算可被修改为如图25所示的计算$e' \leftarrow \alpha(F(r)+e)$。然而，在执行这样的修改的情况中，在d＝0的情况下在图23所示的算法的步骤6中由验证者执行的验证的内容必须由对$c_2 = H_2(\alpha^{-1}(t'-t),t,\alpha^{-1}e'-\alpha F(\alpha^{-1}(t'-t)))$的验证来 取代。

[0675] 如上所述，将在步骤3中由证明者执行的计算$e' \leftarrow \alpha(F(r)-c)+e$可被改变为计算$e' \leftarrow \alpha F(r)+e$(修改A)。此外，在d＝0的情况中将由证明者在步骤5设置的σ的内容可被改变为r,t等(修改B、E)。此外，将在步骤3中由证明者执行的计算$t' \leftarrow \alpha r+t$可被改变为计算$t' \leftarrow \alpha(r+t)$(修改C、F)。此外，将在步骤3中由证明者执行的计算$e' \leftarrow \alpha F(r)+e$可被改变为计算$e' \leftarrow \alpha(F(r)+e)$(修改D、G)。根据修改A的修改、根据修改B和E的修改、根据修改C和F的修改以及根据修改D和G的修改可以分离地被应用于图18所示的高效算法，或者多个修改可以组合地被应用于图18所示的高效算法。

[0676] 在前面，已描述了本发明的第二实施例。另外，多元联立方程式的形式与上述第一实施例的相同。

[0677] <4：高效算法的一般化>

[0678] 现在，根据上述第一和第二实施例的高效算法具有将由以下公式(22)表示的二次多元多项式用作公钥(或系统参数)的结构。然而，上述高效算法可被扩展为具有将在阶q＝p^k的域上定义的三次或更多次的多次多元多项式(参见下面的公式(23))用作公钥(或系统参数)的结构。

[0679] [表达式16]

[0680]

$$f_l(x_1,\cdots,x_n)=\sum_{i=1}^{n}\sum_{j=1}^{n}a_{l,i,j}x_ix_j+\sum_{i=1}^{n}b_{l,i}x_i+c_l \quad \cdots(22)$$

[0681]

$$f_l(x_1,\cdots,x_n)=\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{s=0}^{k-1}\sum_{t=0}^{k-1}a_{l,i,j,s,t}x_i^{p^s}x_j^{p^t}+\sum_{i=1}^{n}\sum_{s=0}^{k-1}b_{l,i,s}x_i^{p^s}+c_l \quad \cdots(23)$$

[0682] 允许在根据上述第一和第二实施例的高效算法中将多元多项式f_1用作公钥的条件是下面的公式(24)相对于(x_1,…,x_n)和(y_1,…,y_n)是双线性的。在由上面的公式(22)表示的多元多项式的情况中，双线性性可以容易地被确认，如下面的公式(25)所示(加下划线的部分分别针对x_1和y_1 是线性的)。此外，与由上面公式(23)表示的多元多项式的情况类似地，双线性性可以如下面的公式(26)所示那样被确认。此外，下面的公式(26)的下划线部分示出了阶p的GF(p)域上的双线性性。因此，在将由上面公式(23)表示的多元多项式用作根据上述第二实施例的高效算法的公钥的情况中，挑战在于在该算法的步骤2之后将由验证者发送的α必须局限于GF(p)的元素。

[0683] [表达式17]

[0684] $f_1(x_1+y_1,\cdots,x_n+y_n)-f_1(x_1,\cdots,x_n)-f_1(y_1,\cdots,y_n)+f_1(0,\cdots,0)$

[0685] …(24)

[0686] $f_l(x_1+y_1,\cdots,x_n+y_n)$

[0687] $=\sum_{i=1}^{n}\sum_{j=1}^{n}a_{l,i,j}(x_i+y_i)(x_j+y_j)+\sum_{i=1}^{n}b_{l,i}(x_i+y_i)+c_l$

[0688] $=\sum_{i=1}^{n}\sum_{j=1}^{n}a_{l,i,j}(x_ix_j+x_iy_j+y_ix_j+y_iy_j)+\sum_{i=1}^{n}b_{l,i}(x_i+y_i)+c_l$

[0689] $=f_l(x_1,\cdots,x_n)+f_l(y_1,\cdots,y_n)+\sum_{i=1}^{n}\sum_{j=1}^{n}a_{l,i,j}\underline{(x_iy_j+y_ix_j)}-f_l(0,\cdots,0)$ …(25)

[0690] $f_l(x_1+y_1,\cdots,x_n+y_n)$

[0691] $$=\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{s=0}^{k-1}\sum_{t=0}^{k-1}a_{l,i,j,s,t}(x_i+y_i)^{p^s}(x_j+y_j)^{p^t}+\sum_{i=1}^{n}\sum_{s=0}^{k-1}b_{l,i,s}(x_i+y_i)^{p^s}+c_l$$

[0692] $$=\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{s=0}^{k-1}\sum_{t=0}^{k-1}a_{l,i,j,s,t}(x_i^{p^s}+y_i^{p^s})(x_j^{p^t}+y_j^{p^t})+\sum_{i=1}^{n}\sum_{s=0}^{k-1}b_{l,i,s}(x_i^{p^s}+y_i^{p^s})+c_l$$

[0693] $$=\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{s=0}^{k-1}\sum_{t=0}^{k-1}a_{l,i,j,s,t}(x_i^{p^s}x_j^{p^t}+x_i^{p^s}y_j^{p^t}+y_i^{p^s}x_j^{p^t}+y_i^{p^s}y_j^{p^t})+\sum_{i=1}^{n}\sum_{s=0}^{k-1}b_{l,i,s}(x_i^{p^s}+y_i^{p^s})+c_l$$

[0694] $$=f_l(x_1,\cdots,x_n)+f_l(y_1,\cdots,y_n)+\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{s=0}^{k-1}\sum_{t=0}^{k-1}a_{l,i,j,s,t}\underline{(x_i^{p^s}y_j^{p^t}+y_i^{p^s}x_j^{p^t})}-f_l(0,\cdots,0)$$

…（26）

[0695] 出于这样的原因，可以说，可以通过扩展根据上述第一和第二实施例的高效算法来构成将由上面公式(23)表示的三次或更多次的多元多项式 用作公钥的算法。

[0696] 这里让我们考虑由上面的公式(22)表示的多元多项式(下面称为二次多项式)与由上面的公式(23)表示的多元多项式(下面称为多次多项式)之间的关系。这里将考虑在阶q＝p的域上定义的nk个变量的二次多项式以及在阶q＝p^k的域上定义的n个变量的多次多项式的考虑。在此情况中，求解由mk个二次多项式形成的联立方程式的难度与求解由m个多次多项式形成的联立方程式的难度是相等的。例如，对于由在阶2的域上定义的80个变量的80个二次多项式形成的联立方程式以及由在阶2^8的域上定义的10个变量的10个多次多项式形成的联立方程式来说，求解难度是相等的。

[0697] 即，如果GF(p^k)的元素和GF(p)^{k}的元素通过同构性(isomorphism)来标识，则将存在由在阶q＝p^k的域上定义的n个变量的m个多次多项式的集合表示的函数，其等同于由在阶q＝p的域上定义的nk个变量的mk个二次多项式的集合表示的函数。例如，如果GF(2^8)的元素和GF(2)^{8}的元素通过同构性来标识，则将存在由在阶2^8的域上定义的10个变量的10个多次多项式的集合表示的函数，其等同于由在阶2的域上定义的80个变量的80个二次多项式的集合表示的函数。因此，可以任意地选择上述二次多项式与上述多次多项式中的哪个将被使用。

[0698] 这里让我们考虑使用二次多项式的情况中的计算效率以及使用上述的多次多项式的情况中的计算效率。

[0699] 在使用在阶2的域上定义的nk个变量的二次多项式的情况中，将针对nk个1比特的变量来执行算法中的操作。即，该操作的单位将为1比特。另一方面，在使用在阶2^k的域上定义的n个变量的多次多项式的情况中，将针对n个k比特的变量来执行算法中的操作。即，该操作的单位为k比特。该k(k＝2,3,4,…)可以任意设置。因此，在实现时，可以通过将k设置为方便的值来提高计算效率。例如，在32比特体系结构上实现算法的情况中，相比于执行基于1比特的操作的配置而言，利用执行基于32比特的操作的配置可以获得更高的计算效率。

[0700] 通过按照使得多次多项式可被用作公钥的以上方式来扩展根据上述第 一和第二实施例的高效算法，可以使操作单位与实现体系结构相匹配。结果，可以提高计算效率。

[0701] 〈5:示例硬件配置〉

[0702] 上述的每个算法例如可以通过利用图26所示的信息处理装置的硬件配置来执行。即，可以通过利用计算机程序控制图26所示的硬件来实现每个算法的处理。另外，该硬件的

模式是任意的，并且可以是个人计算机、诸如移动电话、PHS或PDA之类的移动信息终端、游戏机、接触式或非接触式IC芯片、接触式或非接触式IC卡或者各种类型的信息设备。此外，PHS是是个人手持系统的缩写。此外，PDA是个人数字助理的缩写。

[0703] 如图26所示，该硬件主要包括CPU 902、ROM 904、RAM 906、主机总线908以及桥接器910。此外，该硬件包括外部总线912、接口914、输入单元916、输出单元918、存储单元920、驱动器922、连接端口924以及通信单元926。此外，CPU是中央处理单元的缩写。而且，ROM是只读存储器的缩写。此外，RAM是随机存取存储器的缩写。

[0704] CPU 902例如用作算术处理单元或控制单元，并且基于记录在ROM 904、RAM 906、存储单元920或可移除记录介质928中的各种程序来控制各个构成元件的整体操作或操作的一部分。ROM 904是用于例如存储将被载入到CPU 902上的程序或者在算术运算中使用的数据等的装置。RAM 906例如临时地或永久地存储将被载入到CPU 902上的程序，或者在程序执行时任意改变的各种参数等。

[0705] 这些构成元件例如通过能够执行高速数据传输的主机总线908彼此相连。就此部分而言，例如，主机总线908通过桥接器910连接到外部总线912，外部总线912的数据传输速度相对低。此外，输入单元916例如是鼠标、键盘、触摸面板、按钮、开关或操作杆。此外，输入单元916可以是可利用红外线或其他无线电波发送控制信号的遥控器。

[0706] 输出单元918例如是可以通过视觉或听觉向用户通知所获得的信息的诸如CRT、LCD、PDP或ELD之类的显示设备、诸如扬声器和耳机之类 的音频输出设备、打印机、便携式电话或传真机。此外，CRT是阴极射线管的缩写。LCD是液晶显示器的缩写。PDP是等离子显示面板的缩写。并且，ELD是电致发光显示器的缩写。

[0707] 存储单元920是用于存储各种数据的设备。存储单元920例如是诸如硬盘驱动器(HDD)之类的磁存储设备、半导体存储设备、光存储设备或者磁光存储设备。HDD是硬盘驱动器的缩写。

[0708] 驱动器922是读出记录在诸如磁盘、光盘、磁光盘或半导体存储器之类的可移除记录介质928上的信息，或者将信息写入可移除记录介质928中的设备。可移除记录介质928例如是DVD介质、蓝光介质、HD-DVD介质、各种类型的半导体存储介质等。当然，可移除记录介质928例如可以是安装有非接触式IC芯片的电子设备或IC卡。IC是集成电路的缩写。

[0709] 连接端口924是诸如USB端口、IEEE1394端口、SCSI端口、RS-232C端口之类的端口，或者用于连接诸如光学音频端子之类的外部连接设备930的端口。外部连接设备930例如是打印机、移动音乐播放器、数字相机、数字摄像机或IC记录器。此外，USB是通用串行总线的缩写。并且，SCSI是小型计算机系统接口的缩写。

[0710] 通信设备926是被连接到网络932的通信设备，并且例如是用于有线或无线LAN、蓝牙(注册商标)或者无WUSB的通信卡、光通信路由器、ADSL路由器，或者用于接触式或非接触式通信的设备。连接到通信单元926的网络932是由有线连接或无线连接的网络构成的，并且例如是因特网、家用LAN、红外通信、可见光通信、广播或者卫星通信。此外，LAN是局域网的缩写。而且，WUSB是无线USB的缩写。此外，ADSL是非对称数字订户线的缩写。

[0711] <6：总结>

[0712] 最后，将简要描述根据本发明实施例的技术内容。这里陈述的技术内容可被应用于各种信息处理装置，例如个人计算机、移动电话、便携式游戏机、便携式信息终端、信息设

备、汽车导航系统等。

[0713] 上面的信息处理装置的功能配置可被表述为如下。该信息处理装置包括如下的密钥设置单元、消息发送单元、验证模式接收单元以及响应发送 单元。密钥设置单元用于将$s \in K^n$设置为私钥并且将环K上的多次多项式$f_i(x_1,\cdots,x_n)$(i＝1至m)以及$y_i = f_i(s)$设置为公钥。针对i＝1至m来找出$y_i = f_i(s)$的问题正是求解多次多元联立方程式的问题。即，上述信息处理装置用于提供将求解多次多元联立方程式的问题当作安全性的基础的认证机制。

[0714] 并且，消息发送单元用于将消息c发送给验证者。此外，验证模式接收单元用于接收与验证者针对一个消息c从k(k≥3)个验证模式中选择的一个验证模式有关的信息。此外，响应发送单元用于向验证者发送k种响应信息中的与由验证模式接收单元接收到的验证模式有关的信息所对应的响应信息。此外，响应信息是这样的信息，该信息使得在通过利用k种响应信息执行的针对消息c的所有k个验证模式都成功的情况中能够计算出私钥s。

[0715] 消息c例如包括三种消息c_1，c_2和c_3。此外，针对c_1，c_2和c_3中的任意两者的验证通过每种验证模式来执行。此外，响应信息是根据每种验证模式执行针对两种消息的验证所需的信息。此外，一个条件是：即使消息c被传递给验证者，与私钥有关的信息也不会被泄露。另一条件是：即使消息c和响应信息两者被提供，与私钥有关的信息也不会被泄露。满足这些条件的消息c的结构和响应信息的结构已经参考图4等进行了具体描述。

[0716] (注释)

[0717] 密钥生成算法Gen是密钥设置单元的一个示例。证明者算法P是消息发送单元、验证模式接收单元、响应发送单元、答复接收单元、多项式生成单元以及多项式发送单元的示例。信息σ是响应信息的示例。验证者算法V是消息接收单元、验证模式选择单元、验证模式发送单元、响应接收单元、验证单元、答复发送单元和多项式接收单元的示例。签名生成算法Sig是消息生成单元、验证模式选择单元以及签名生成单元的示例。

[0718] 本领域的技术人员应当明白，可以根据设计要求和其它因素进行各种修改、组合、子组合和变更，只要它们在所附权利要求或其等同物的范围之内。

[0719] 本申请包含与分别于2010年5月31日、2010年10月4日和2011年 2月9日向日本专利局提交的日本优先专利申请2010-125026、JP 2010-224753和JP 2011-026401中公开的主题有关的主题，该申请的全部内容通过引用结合于此。

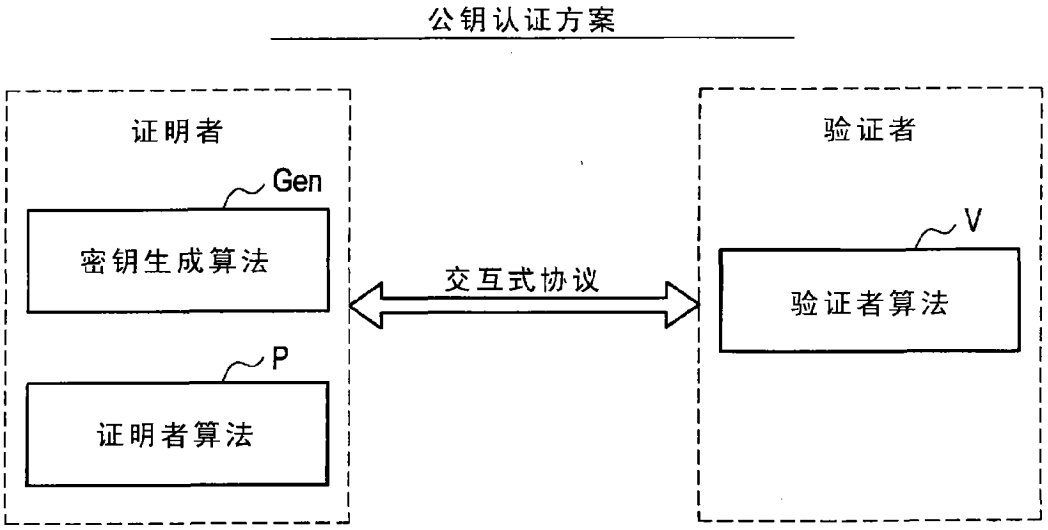

公钥认证方案
证明者
Gen
密钥生成算法
P
证明者算法
交互式协议
验证者
V
验证者算法

图1

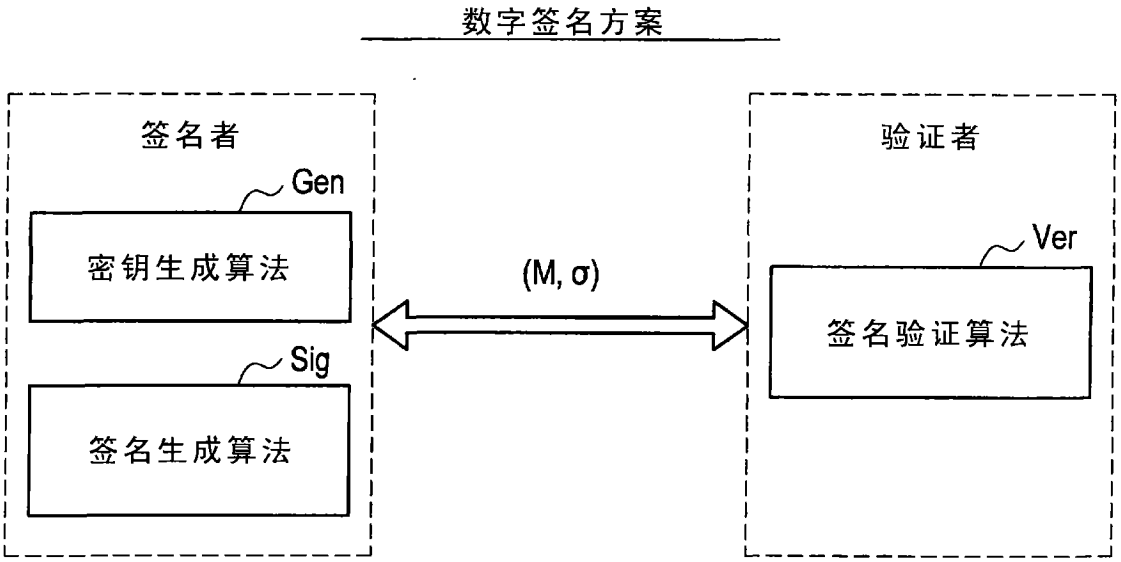

数字签名方案
签名者
Gen
密钥生成算法
Sig
签名生成算法
(M, σ)
验证者
Ver
签名验证算法

图2

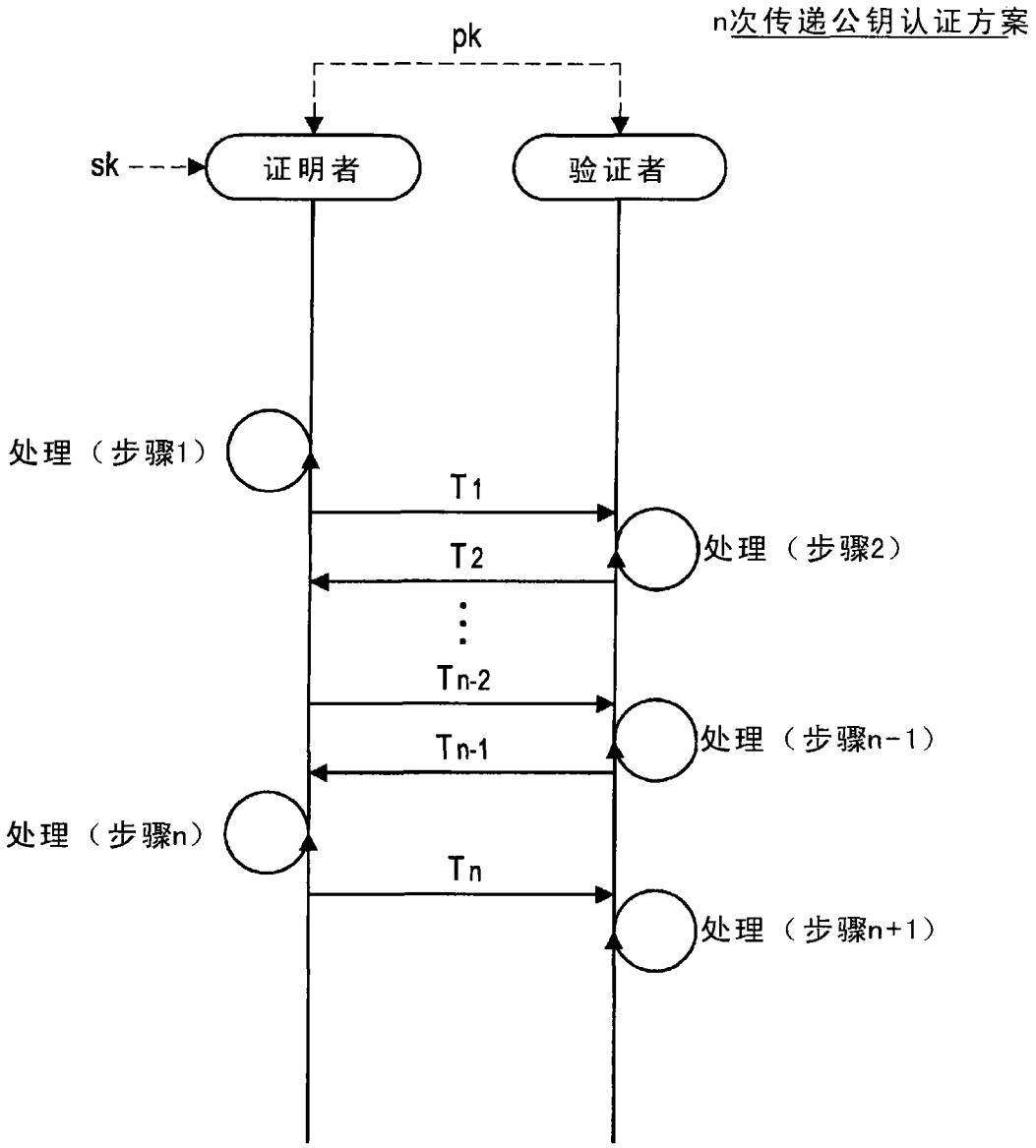

n次传递公钥认证方案
pk
sk
证明者
验证者
处理（步骤1）
T1
处理（步骤2）
T2
Tn-2
处理（步骤n-1）
Tn-1
处理（步骤n）
Tn
处理（步骤n+1）

图3

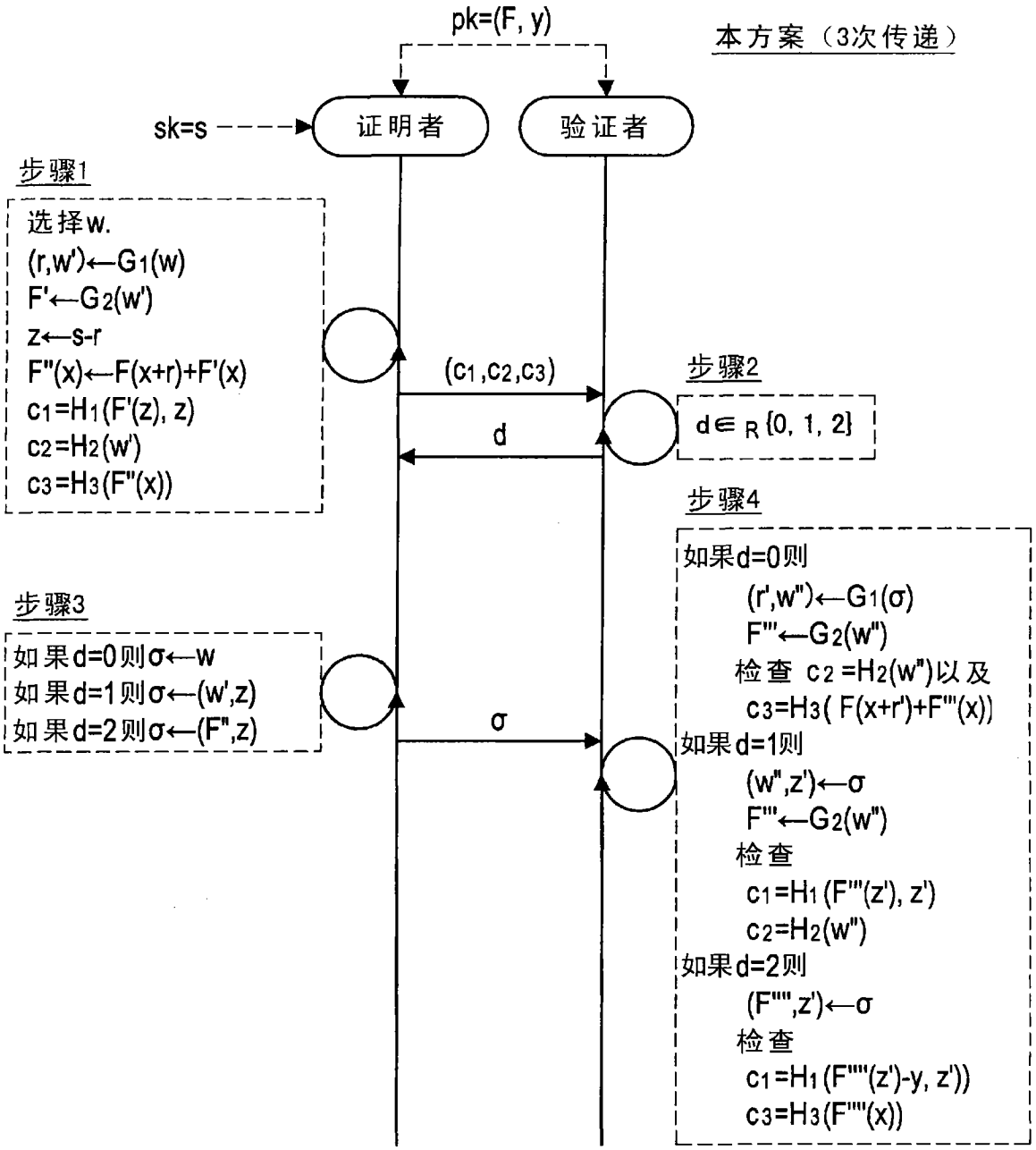

pk=(F, y)
本方案（3次传递）
sk=s
证明者
验证者
步骤1
选择w.
(r,w')←G1(w)
F'←G2(w')
z←s-r
F''(x)←F(x+r)+F'(x)
c1=H1(F'(z), z)
c2=H2(w')
c3=H3(F''(x))
(c1,c2,c3)
步骤2
d∈R{0, 1, 2}
d
步骤4
如果d=0则
(r',w'')←G1(σ)
F'''←G2(w'')
检查 c2=H2(w'')以及
c3=H3(F(x+r')+F'''(x))
如果d=1则
(w'',z')←σ
F'''←G2(w'')
检查
c1=H1(F'''(z'), z')
c2=H2(w'')
如果d=2则
(F'''',z')←σ
检查
c1=H1(F''''(z')-y, z'))
c3=H3(F''''(x))
步骤3
如果d=0则σ←w
如果d=1则σ←(w',z)
如果d=2则σ←(F'',z)
σ

图4

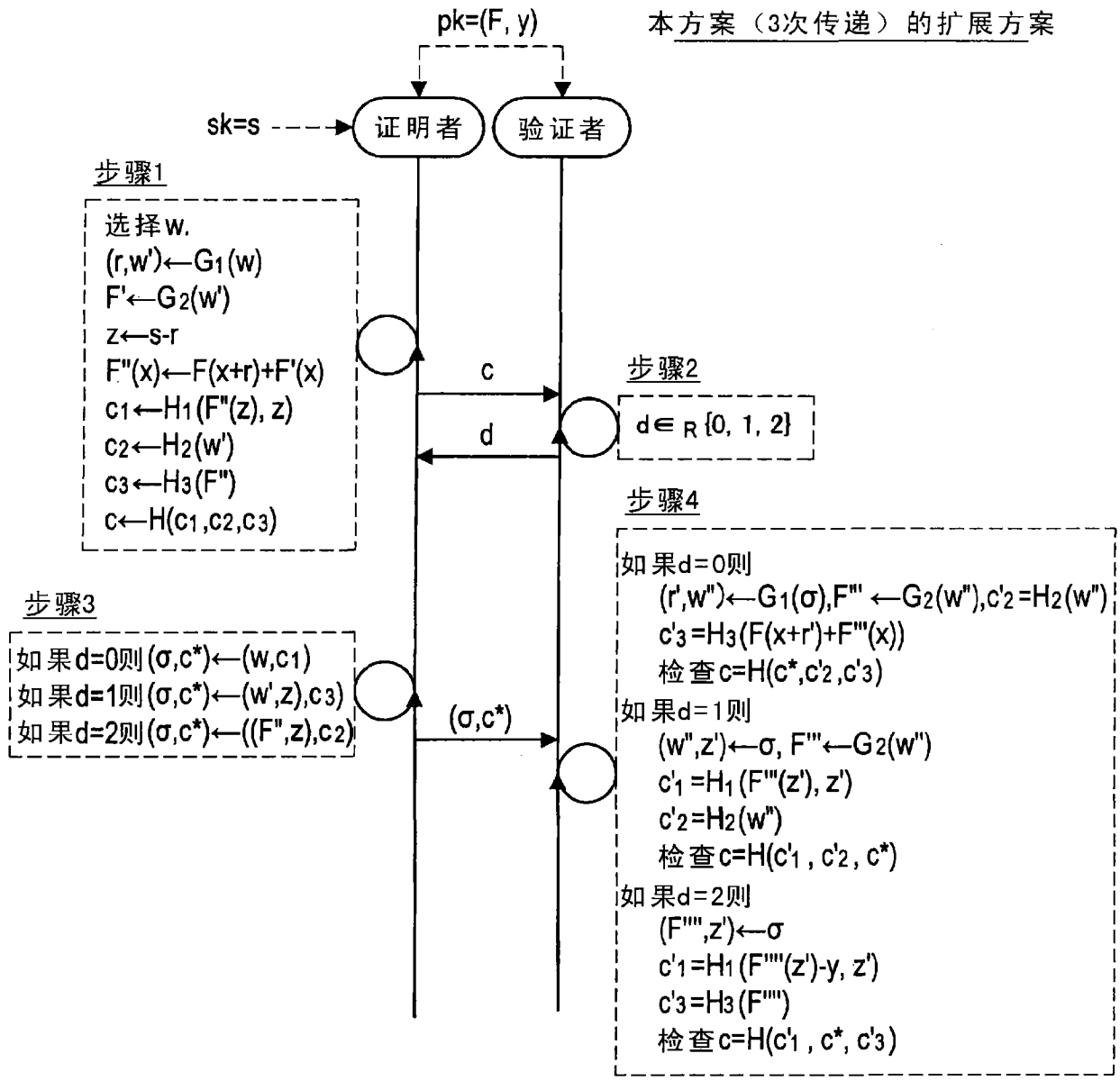

pk=(F, y)
本方案（3次传递）的扩展方案
sk=s
证明者
验证者
步骤1
选择w.
(r,w')←G1(w)
F'←G2(w')
z←s-r
F''(x)←F(x+r)+F'(x)
c1←H1(F''(z), z)
c2←H2(w')
c3←H3(F'')
c←H(c1,c2,c3)
c
d
步骤2
d∈R{0, 1, 2}
步骤3
如果d=0则(σ,c*)←(w,c1)
如果d=1则(σ,c*)←(w',z),c3)
如果d=2则(σ,c*)←((F'',z),c2)
(σ,c*)
步骤4
如果d=0则
(r',w'')←G1(σ),F''' ←G2(w''),c'2=H2(w'')
c'3=H3(F(x+r')+F'''(x))
检查c=H(c*,c'2,c'3)
如果d=1则
(w'',z')←σ, F'''←G2(w'')
c'1=H1(F'''(z'), z')
c'2=H2(w'')
检查c=H(c'1, c'2, c*)
如果d=2则
(F'''',z')←σ
c'1=H1(F''''(z')-y, z')
c'3=H3(F'''')
检查c=H(c'1, c*, c'3)

图5

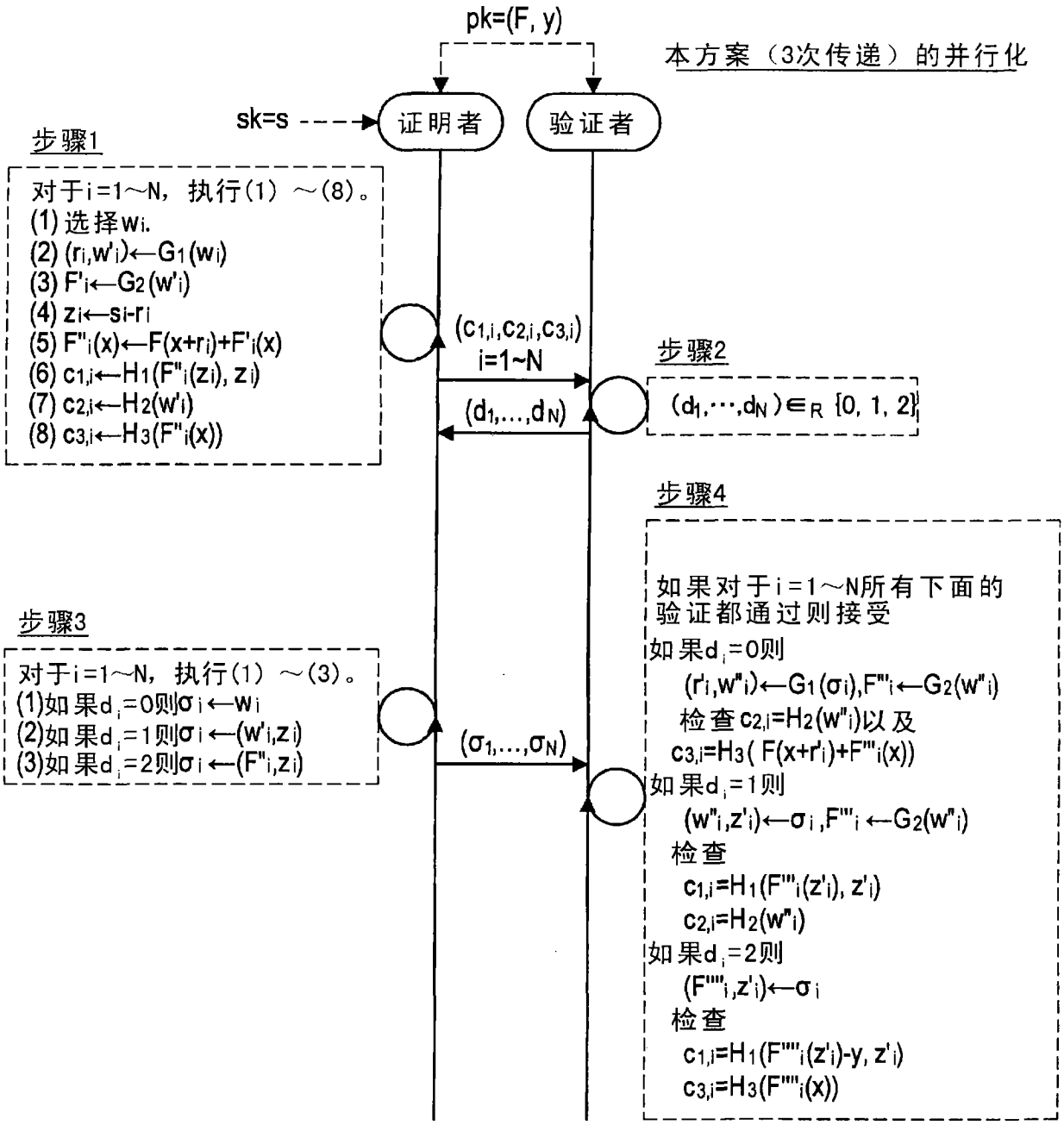

pk=(F, y)
本方案（3次传递）的并行化
sk=s
证明者
验证者
步骤1
对于i=1～N，执行(1)～(8)。
(1) 选择wi.
(2) (ri,w'i)←G1(wi)
(3) F'i←G2(w'i)
(4) zi←si-ri
(5) F"i(x)←F(x+ri)+F'i(x)
(6) c1,i←H1(F"i(zi), zi)
(7) c2,i←H2(w'i)
(8) c3,i←H3(F"i(x))
(c1,i,c2,i,c3,i)
i=1~N
步骤2
(d1,…,dN)∈R {0, 1, 2}
(d1,…,dN)
步骤3
对于i=1～N，执行(1)～(3)。
(1)如果di=0则σi←wi
(2)如果di=1则σi←(w'i,zi)
(3)如果di=2则σi←(F"i,zi)
(σ1,…,σN)
步骤4
如果对于i=1～N所有下面的验证都通过则接受
如果di=0则
(r'i,w"i)←G1(σi),F"'i←G2(w"i)
检查c2,i=H2(w"i)以及
c3,i=H3(F(x+r'i)+F"'i(x))
如果di=1则
(w"i,z'i)←σi,F"'i←G2(w"i)
检查
c1,i=H1(F"'i(z'i), z'i)
c2,i=H2(w"i)
如果di=2则
(F""i,z'i)←σi
检查
c1,i=H1(F""i(z'i)-y, z'i)
c3,i=H3(F""i(x))

图6

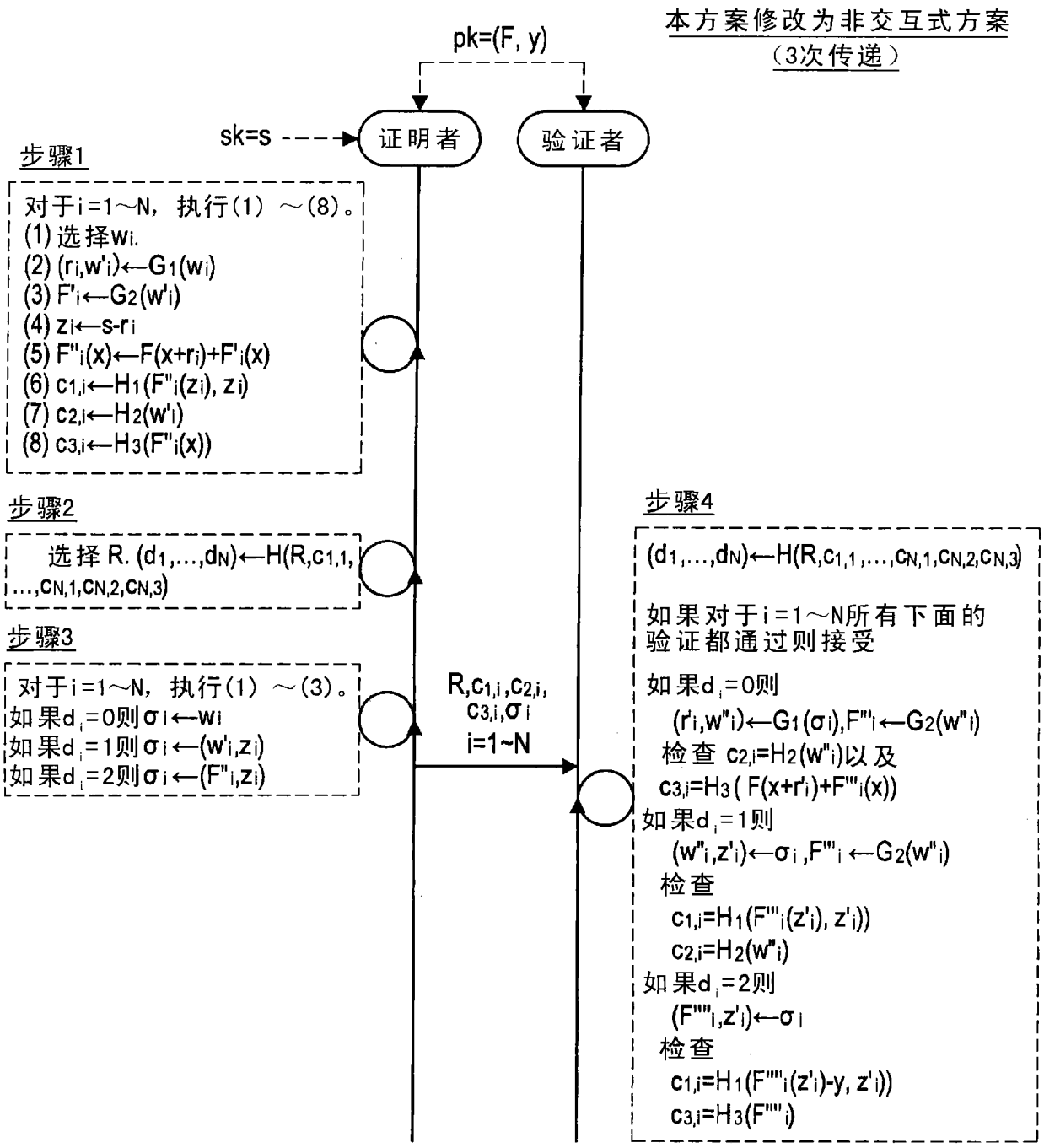

本方案修改为非交互式方案
（3次传递）
pk=(F, y)
sk=s
证明者
验证者
步骤1
对于i=1～N，执行(1)～(8)。
(1) 选择wi.
(2) (ri,w'i)←G1(wi)
(3) F'i←G2(w'i)
(4) zi←s-ri
(5) F"i(x)←F(x+ri)+F'i(x)
(6) c1,i←H1(F"i(zi), zi)
(7) c2,i←H2(w'i)
(8) c3,i←H3(F"i(x))
步骤2
选择 R. (d1,...,dN)←H(R,c1,1,...,cN,1,cN,2,cN,3)
步骤3
对于i=1～N，执行(1)～(3)。
如果di=0则σi←wi
如果di=1则σi←(w'i,zi)
如果di=2则σi←(F"i,zi)
R,c1,i,c2,i,c3,i,σi
i=1~N
步骤4
(d1,...,dN)←H(R,c1,1,...,cN,1,cN,2,cN,3)
如果对于i=1～N所有下面的验证都通过则接受
如果di=0则
(r'i,w"i)←G1(σi),F"'i←G2(w"i)
检查 c2,i=H2(w"i)以及
c3,i=H3(F(x+r'i)+F"'i(x))
如果di=1则
(w"i,z'i)←σi,F"'i←G2(w"i)
检查
c1,i=H1(F"'i(z'i), z'i))
c2,i=H2(w"i)
如果di=2则
(F""i,z'i)←σi
检查
c1,i=H1(F""i(z'i)-y, z'i))
c3,i=H3(F""i)

图7

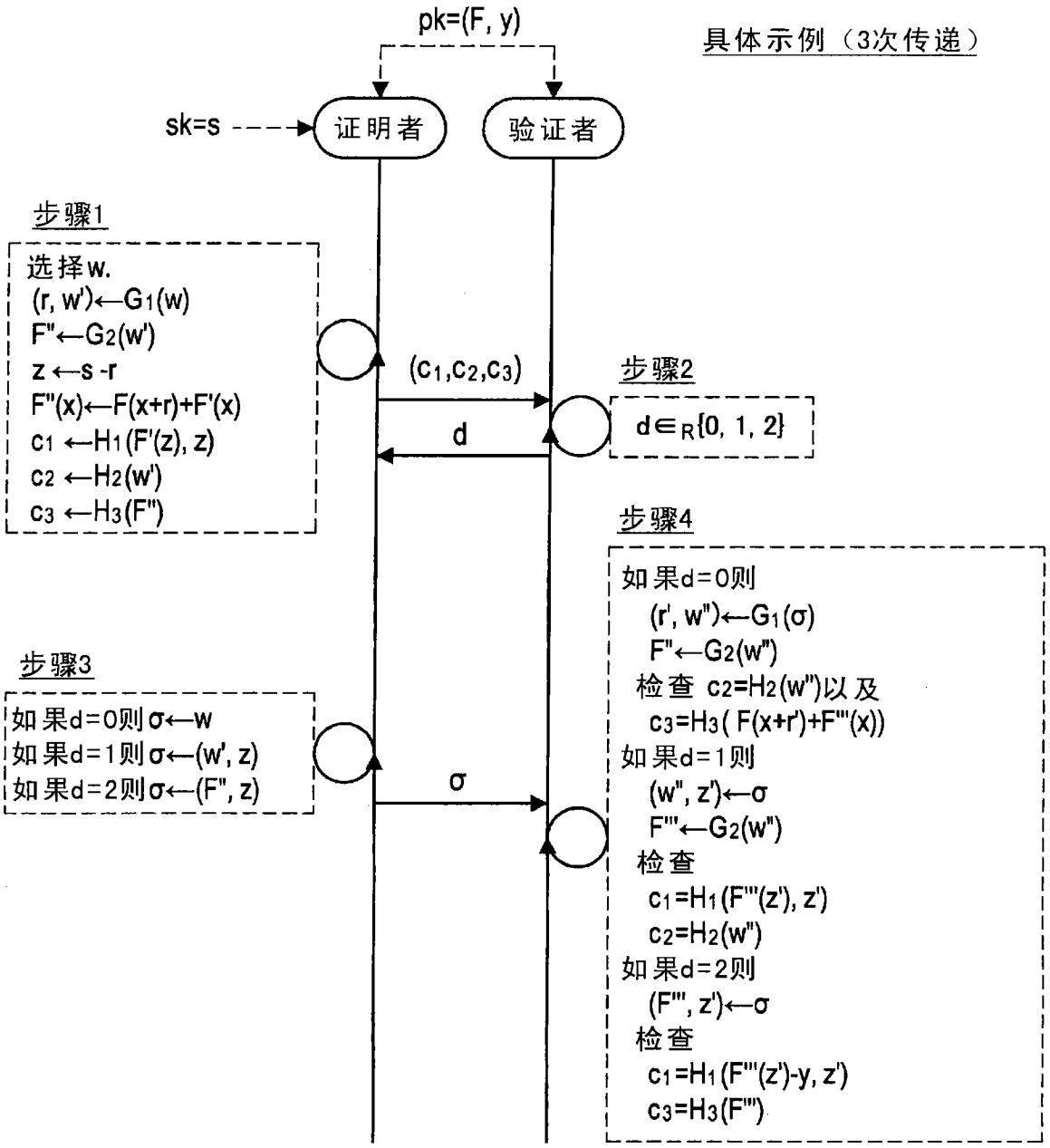

pk=(F, y)
具体示例（3次传递）
sk=s
证明者
验证者
步骤1
选择w.
(r, w')←G1(w)
F"←G2(w')
z←s -r
F"(x)←F(x+r)+F'(x)
c1←H1(F'(z), z)
c2←H2(w')
c3←H3(F")
(c1,c2,c3)
步骤2
d∈R{0, 1, 2}
d
步骤4
如果d=0则
(r', w")←G1(σ)
F"←G2(w")
检查 c2=H2(w")以及
c3=H3(F(x+r')+F"'(x))
如果d=1则
(w", z')←σ
F"'←G2(w")
检查
c1=H1(F"'(z'), z')
c2=H2(w")
如果d=2则
(F"', z')←σ
检查
c1=H1(F"'(z')-y, z')
c3=H3(F"')
步骤3
如果d=0则σ←w
如果d=1则σ←(w', z)
如果d=2则σ←(F", z)
σ

图8

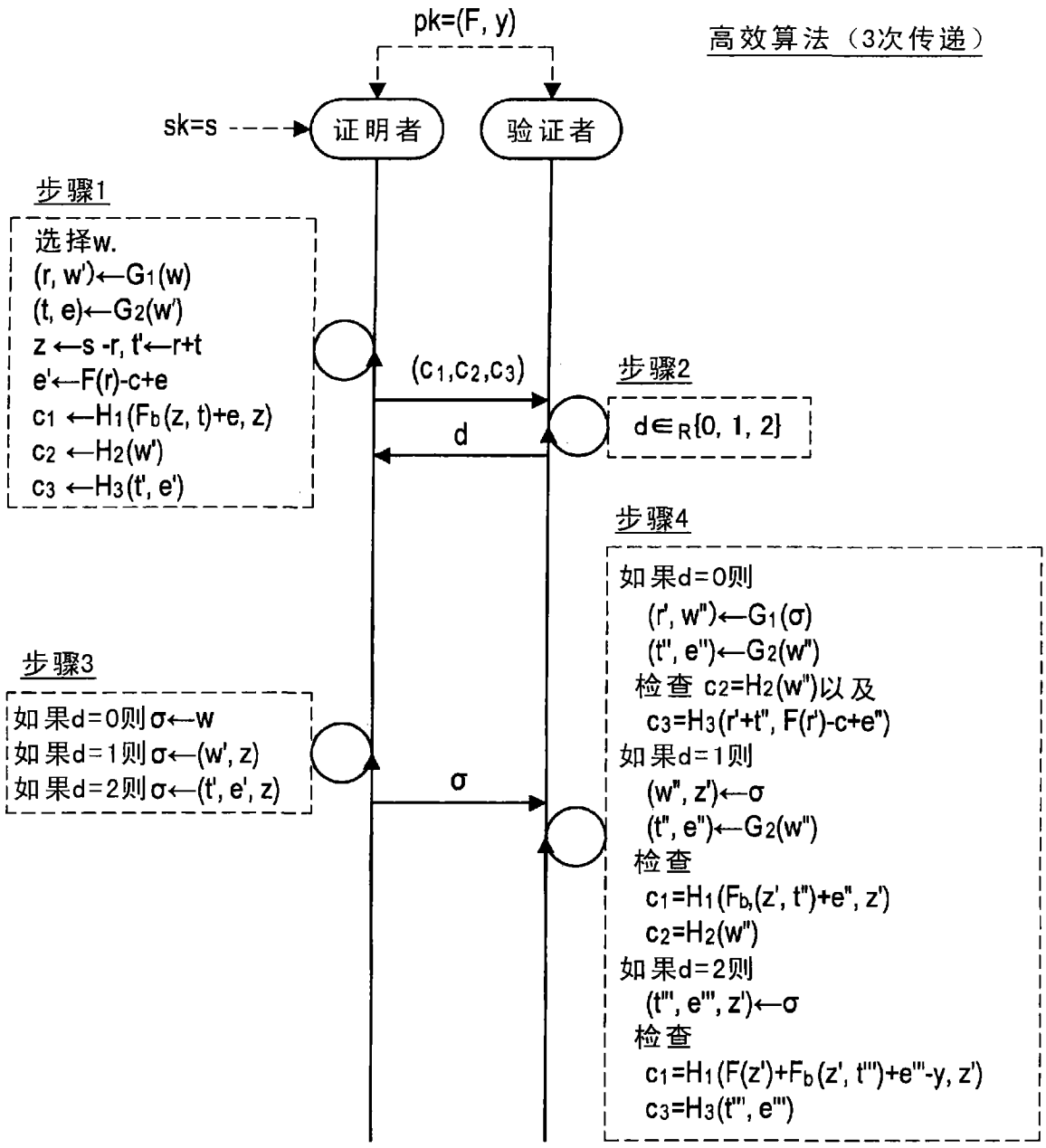

pk=(F, y)
高效算法（3次传递）
sk=s
证明者
验证者
步骤1
选择w.
(r, w')←G1(w)
(t, e)←G2(w')
z←s -r, t'←r+t
e'←F(r)-c+e
c1←H1(Fb(z, t)+e, z)
c2←H2(w')
c3←H3(t', e')
(c1,c2,c3)
步骤2
d∈R{0, 1, 2}
d
步骤4
如果d=0则
(r', w")←G1(σ)
(t", e")←G2(w")
检查 c2=H2(w")以及
c3=H3(r'+t", F(r')-c+e")
如果d=1则
(w", z')←σ
(t", e")←G2(w")
检查
c1=H1(Fb,(z', t")+e", z')
c2=H2(w")
如果d=2则
(t"', e"', z')←σ
检查
c1=H1(F(z')+Fb(z', t"')+e"'-y, z')
c3=H3(t"', e"')
步骤3
如果d=0则σ←w
如果d=1则σ←(w', z)
如果d=2则σ←(t', e', z)
σ

图9

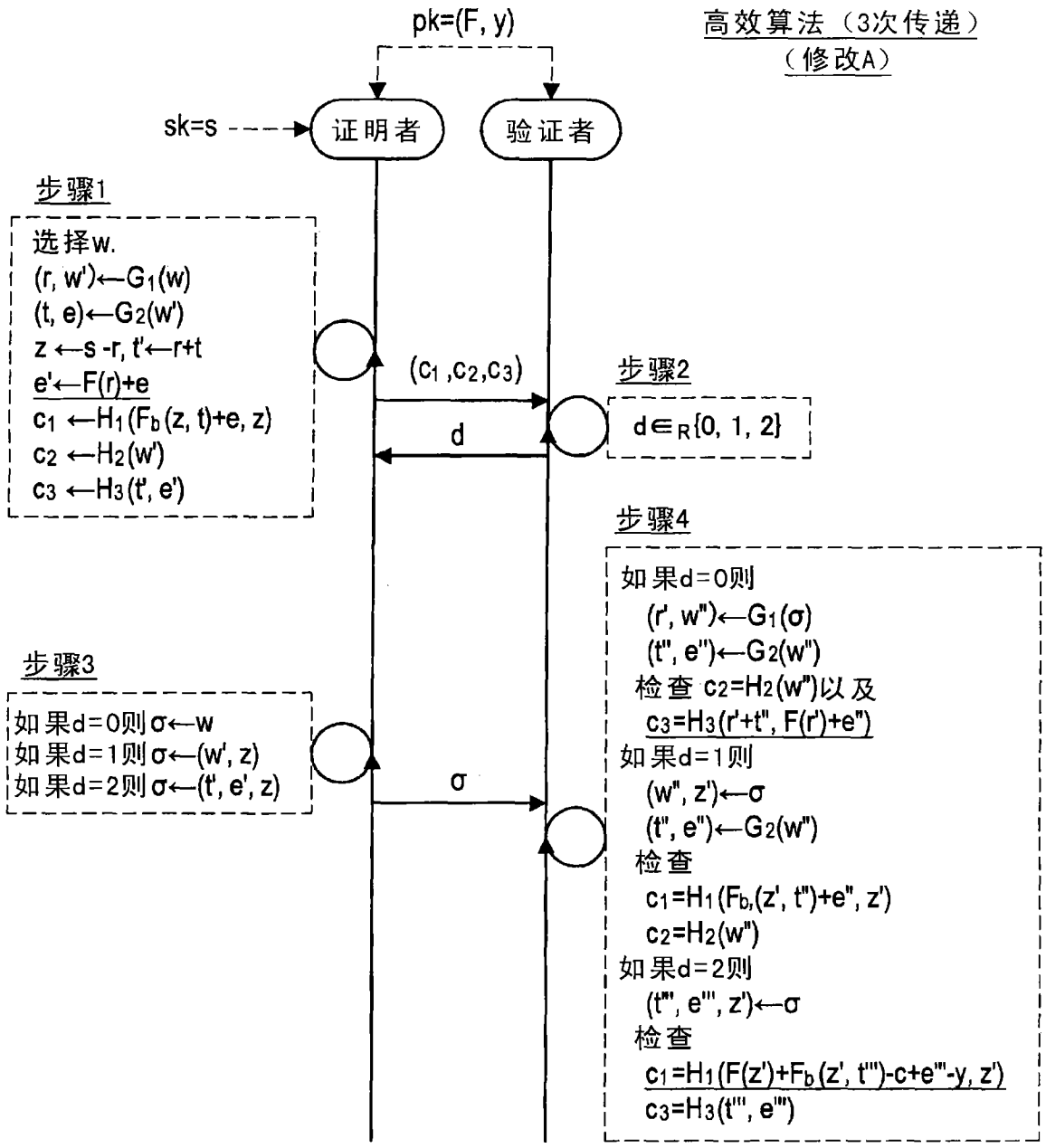

pk=(F, y)
高效算法（3次传递）
（修改A）
sk=s
证明者
验证者
步骤1
选择w.
(r, w')←G1(w)
(t, e)←G2(w')
z ←s -r, t'←r+t
e'←F(r)+e
c1 ←H1(Fb(z, t)+e, z)
c2 ←H2(w')
c3 ←H3(t', e')
(c1 ,c2,c3)
步骤2
d∈R{0, 1, 2}
d
步骤4
如果d=0则
(r', w")←G1(σ)
(t", e")←G2(w")
检查 c2=H2(w")以及
c3=H3(r'+t", F(r')+e")
如果d=1则
(w", z')←σ
(t", e")←G2(w")
检查
c1=H1(Fb,(z', t")+e", z')
c2=H2(w")
如果d=2则
(t"', e"', z')←σ
检查
c1=H1(F(z')+Fb(z', t"')-c+e"'-y, z')
c3=H3(t"', e"')
步骤3
如果d=0则σ←w
如果d=1则σ←(w', z)
如果d=2则σ←(t', e', z)
σ

图10

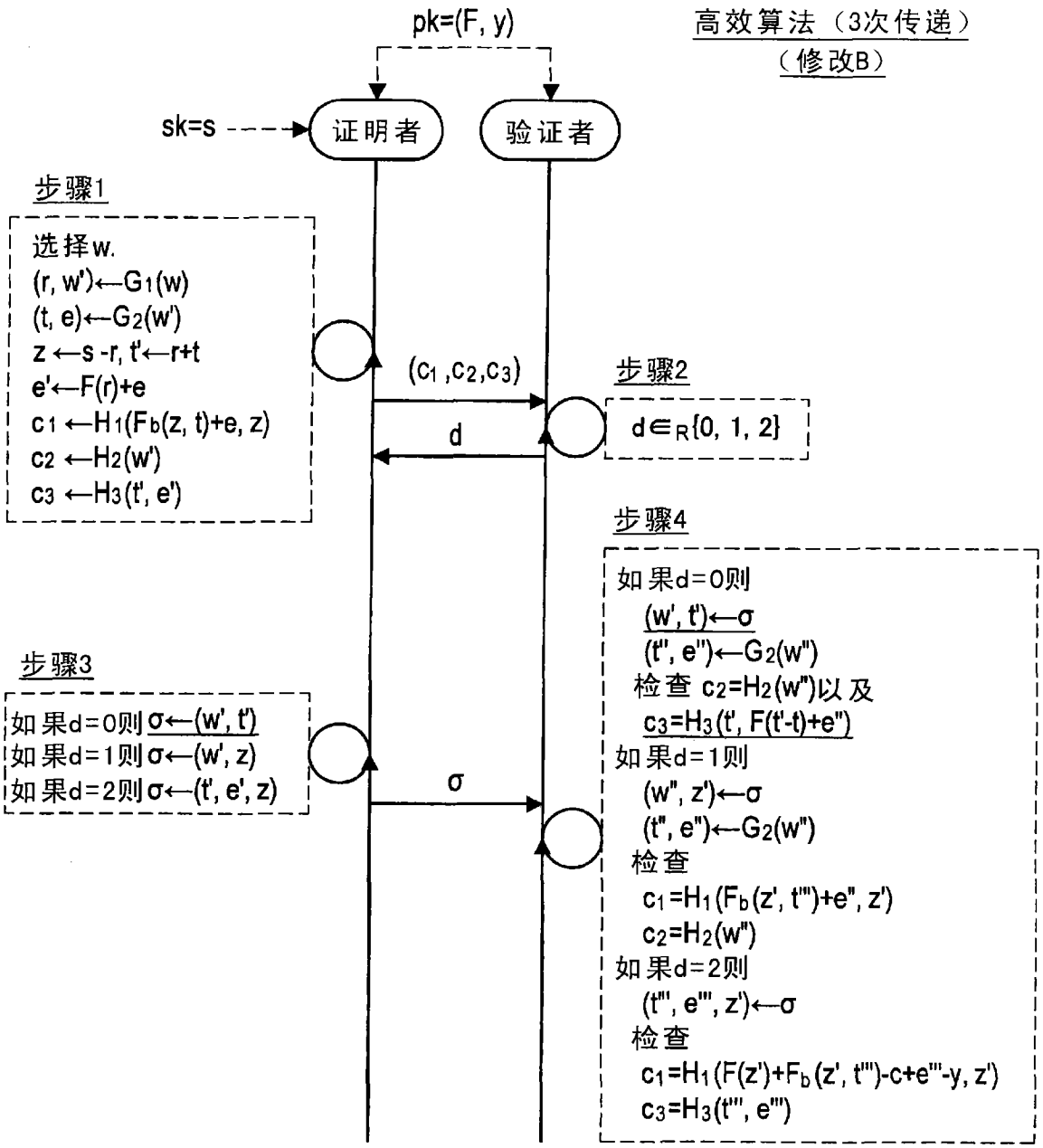

pk=(F, y)
高效算法（3次传递）
（修改B）
sk=s
证明者
验证者
步骤1
选择w.
(r, w')←G1(w)
(t, e)←G2(w')
z←s -r, t'←r+t
e'←F(r)+e
c1←H1(Fb(z, t)+e, z)
c2←H2(w')
c3←H3(t', e')
(c1 ,c2,c3)
步骤2
d∈R{0, 1, 2}
d
步骤4
如果d=0则
(w', t')←σ
(t", e")←G2(w")
检查 c2=H2(w")以及
c3=H3(t', F(t'-t)+e")
如果d=1则
(w", z')←σ
(t", e")←G2(w")
检查
c1=H1(Fb(z', t"')+e", z')
c2=H2(w")
如果d=2则
(t"', e"', z')←σ
检查
c1=H1(F(z')+Fb(z', t"')-c+e"'-y, z')
c3=H3(t"', e"')
步骤3
如果d=0则σ←(w', t')
如果d=1则σ←(w', z)
如果d=2则σ←(t', e', z)
σ

图11

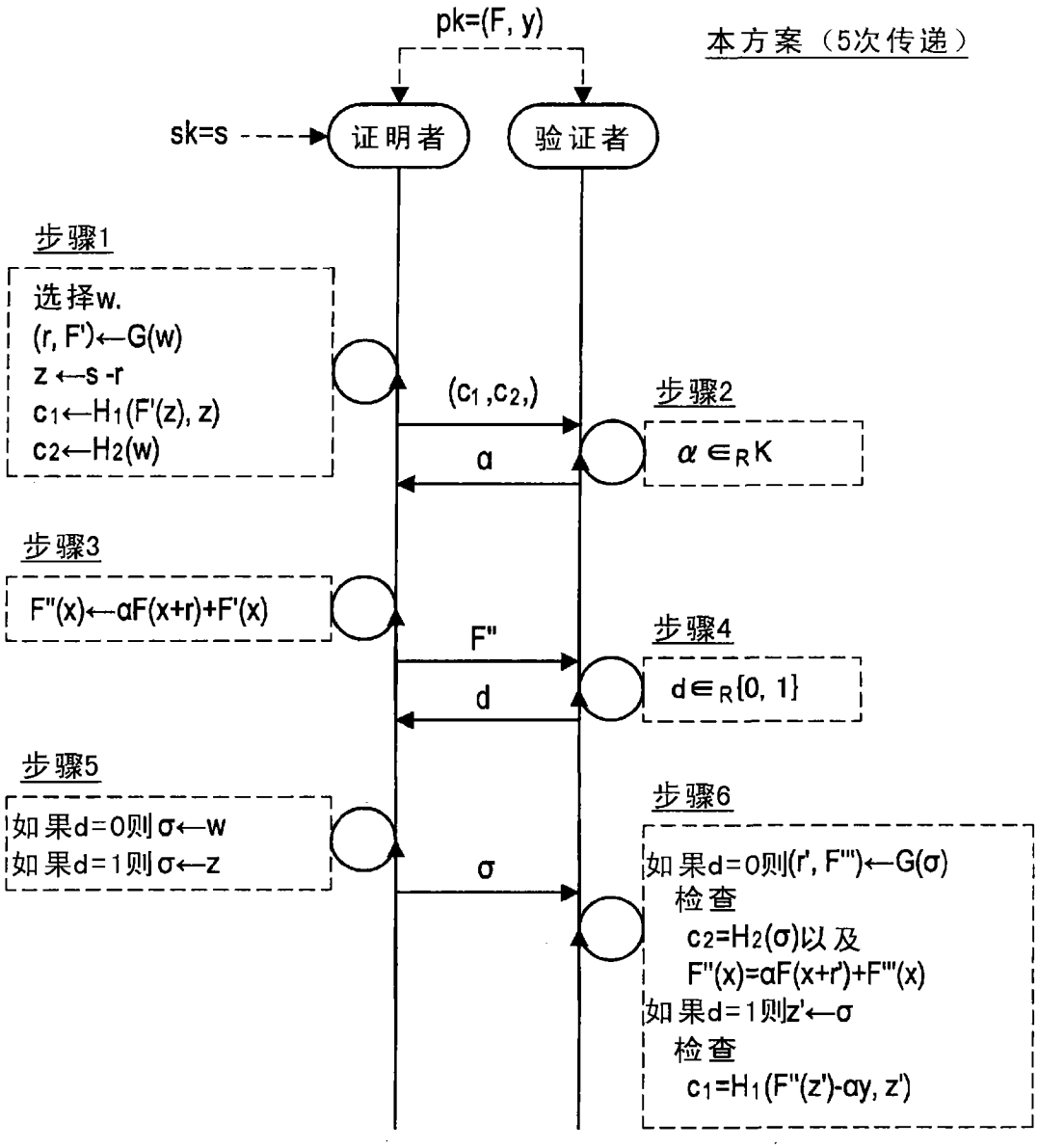

pk=(F, y)
本方案（5次传递）
sk=s
证明者
验证者
步骤1
选择w.
(r, F')←G(w)
z←s -r
c1←H1(F'(z), z)
c2←H2(w)
(c1 ,c2,)
步骤2
α
α ∈R K
步骤3
F''(x)←αF(x+r)+F'(x)
F''
步骤4
d
d∈R{0, 1}
步骤5
如果d=0则σ←w
如果d=1则σ←z
步骤6
σ
如果d=0则(r', F''')←G(σ)
检查
c2=H2(σ)以及
F''(x)=αF(x+r')+F'''(x)
如果d=1则z'←σ
检查
c1=H1(F''(z')-αy, z')

图12

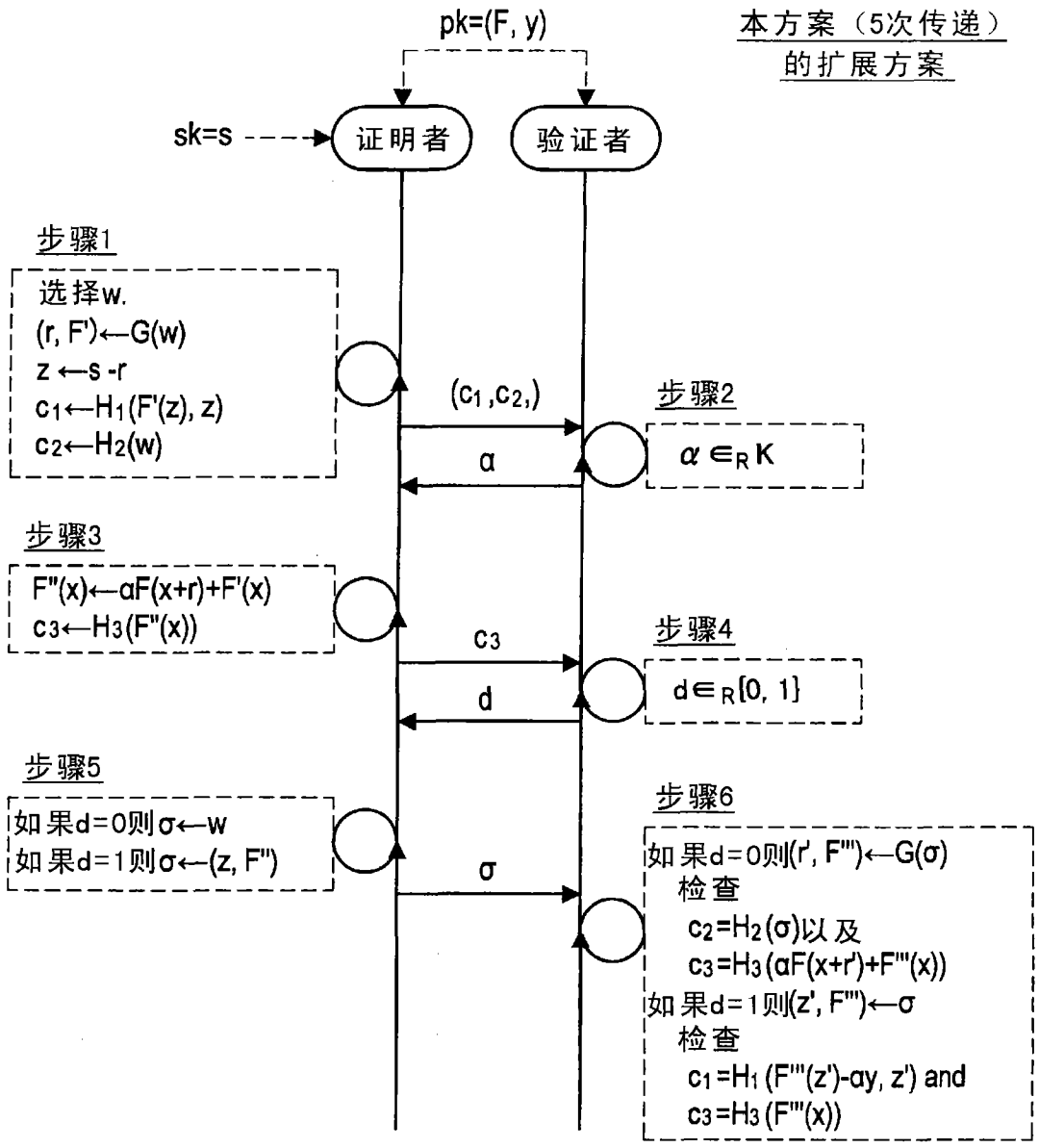

pk=(F, y)
本方案（5次传递）
的扩展方案
sk=s
证明者
验证者
步骤1
选择w.
(r, F')←G(w)
z ←s -r
c1←H1(F'(z), z)
c2←H2(w)
(c1 ,c2,)
步骤2
α
α ∈R K
步骤3
F"(x)←αF(x+r)+F'(x)
c3←H3(F"(x))
c3
步骤4
d
d∈R{0, 1}
步骤5
如果d=0则σ←w
如果d=1则σ←(z, F")
σ
步骤6
如果d=0则(r', F"')←G(σ)
检查
c2=H2(σ)以及
c3=H3(αF(x+r')+F"'(x))
如果d=1则(z', F"')←σ
检查
c1=H1(F"'(z')-αy, z') and
c3=H3(F"'(x))

图13

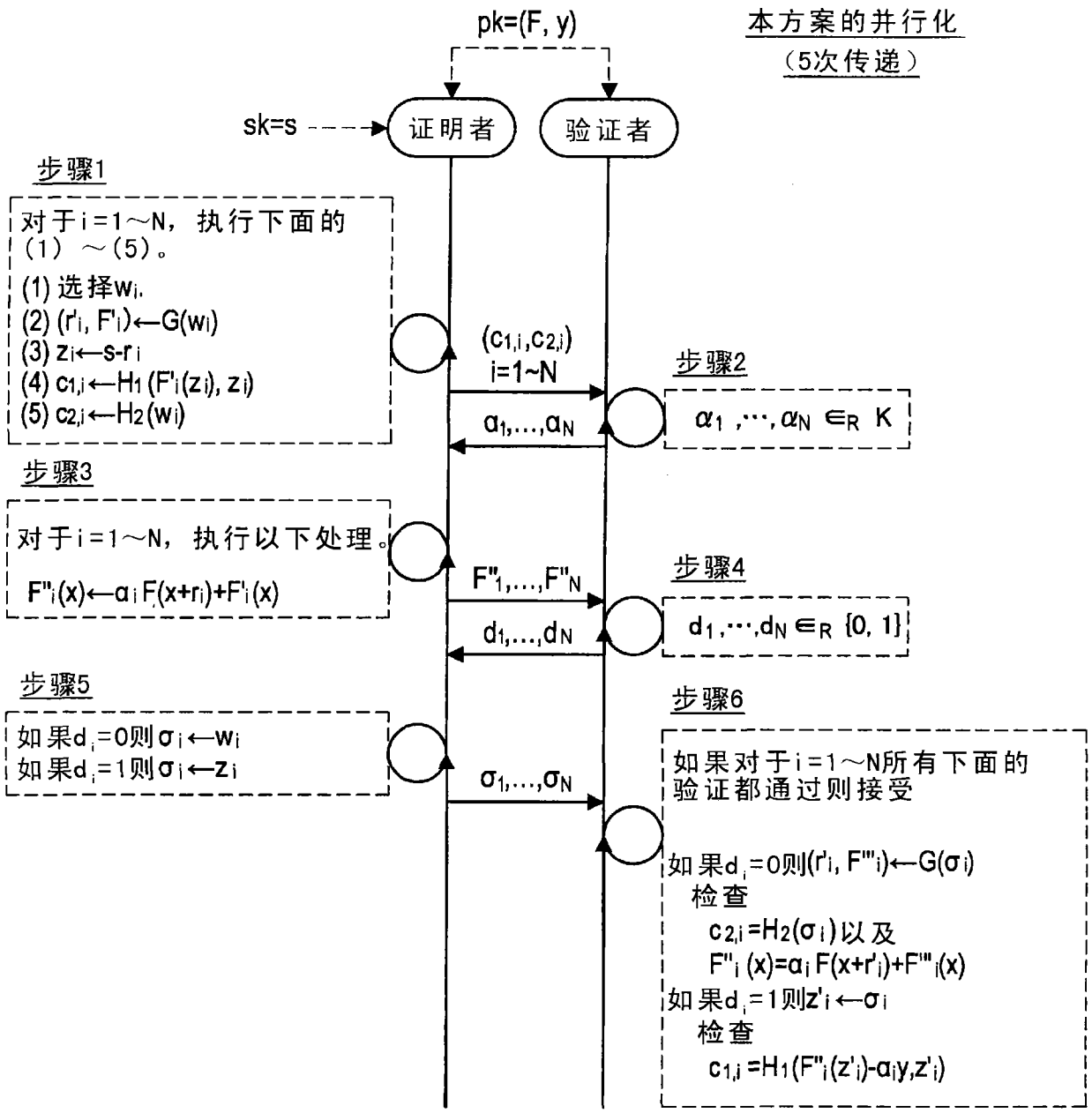

pk=(F, y)
本方案的并行化
（5次传递）
sk=s
证明者
验证者
步骤1
对于i=1～N，执行下面的（1）～(5)。
(1) 选择wi.
(2) (r'i, F'i)←G(wi)
(3) zi←s-r i
(4) c1,i←H1(F'i(zi), zi)
(5) c2,i←H2(wi)
(c1,i,c2,i) i=1~N
步骤2
α1 ,⋯, αN ∈R K
α1,...,αN
步骤3
对于i=1～N，执行以下处理。
F"i(x)←αi F(x+ri)+F'i(x)
F"1,...,F"N
步骤4
d1 ,⋯,dN ∈R {0, 1}
d1,...,dN
步骤5
如果di=0则σi←wi
如果di=1则σi←zi
σ1,...,σN
步骤6
如果对于i=1～N所有下面的验证都通过则接受
如果di=0则(r'i, F'"i)←G(σi)
检查
c2,i=H2(σi)以及
F"i (x)=αi F(x+r'i)+F'"i(x)
如果di=1则z'i←σi
检查
c1,i=H1(F"i(z'i)-αiy,z'i)

图14

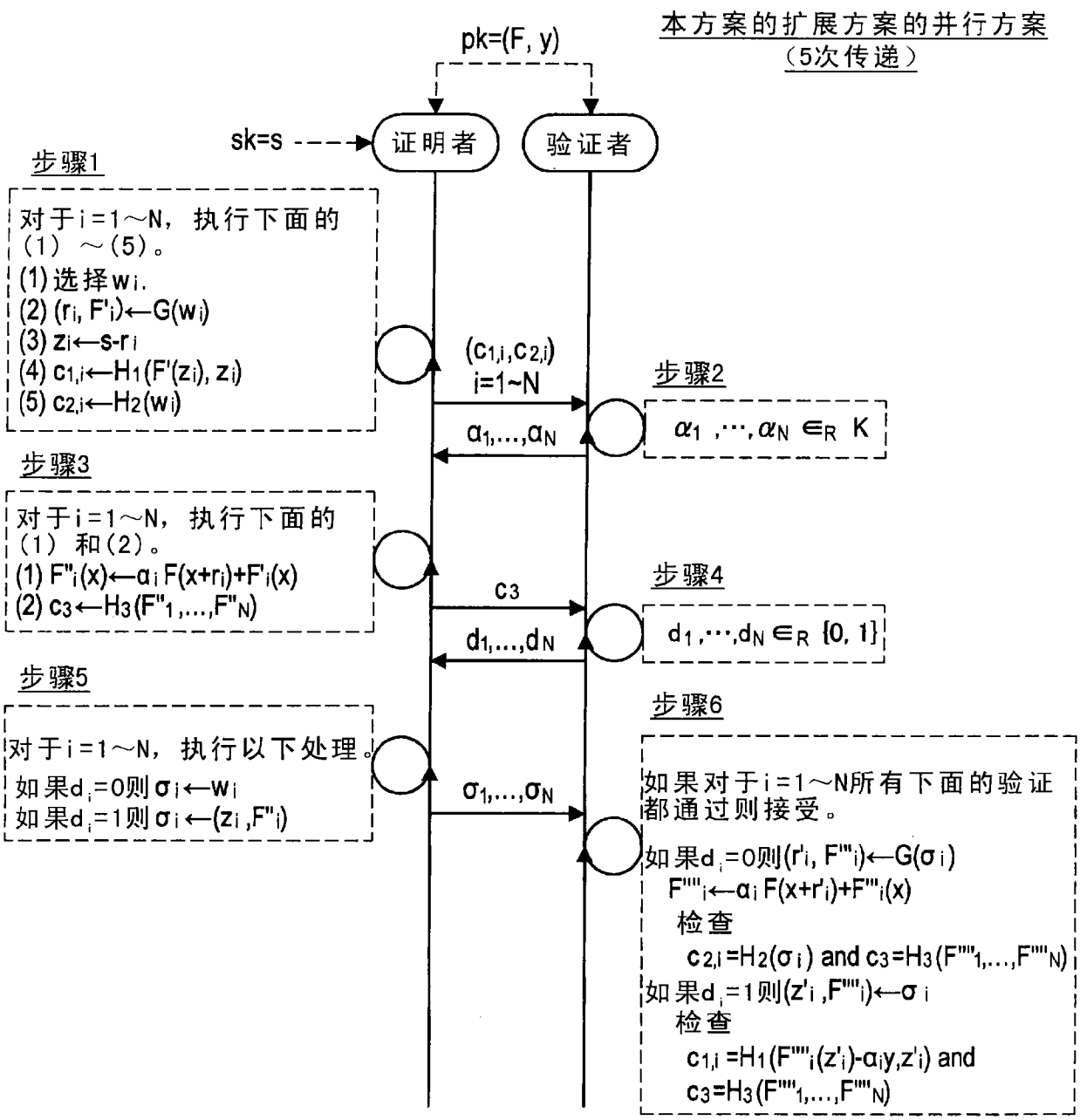

本方案的扩展方案的并行方案
（5次传递）
pk=(F, y)
sk=s
证明者
验证者
步骤1
对于i=1～N，执行下面的（1）～(5)。
(1) 选择wi.
(2) (ri, F'i)←G(wi)
(3) zi←s-ri
(4) c1,i←H1(F'(zi), zi)
(5) c2,i←H2(wi)
(c1,i,c2,i) i=1~N
步骤2
α1 ,⋯,αN ∈R K
α1,...,αN
步骤3
对于i=1～N，执行下面的（1）和(2)。
(1) F"i(x)←αi F(x+ri)+F'i(x)
(2) c3←H3(F"1,...,F"N)
c3
步骤4
d1 ,⋯,dN ∈R {0, 1}
d1,...,dN
步骤5
对于i=1～N，执行以下处理。
如果di=0则σi←wi
如果di=1则σi←(zi ,F"i)
σ1,...,σN
步骤6
如果对于i=1～N所有下面的验证都通过则接受。
如果di=0则(r'i, F'"i)←G(σi)
F""i←αi F(x+r'i)+F'"i(x)
检查
c2,i=H2(σi) and c3=H3(F""1,...,F""N)
如果di=1则(z'i ,F""i)←σi
检查
c1,i =H1(F""i(z'i)-αiy,z'i) and
c3=H3(F""1,...,F""N)

图15

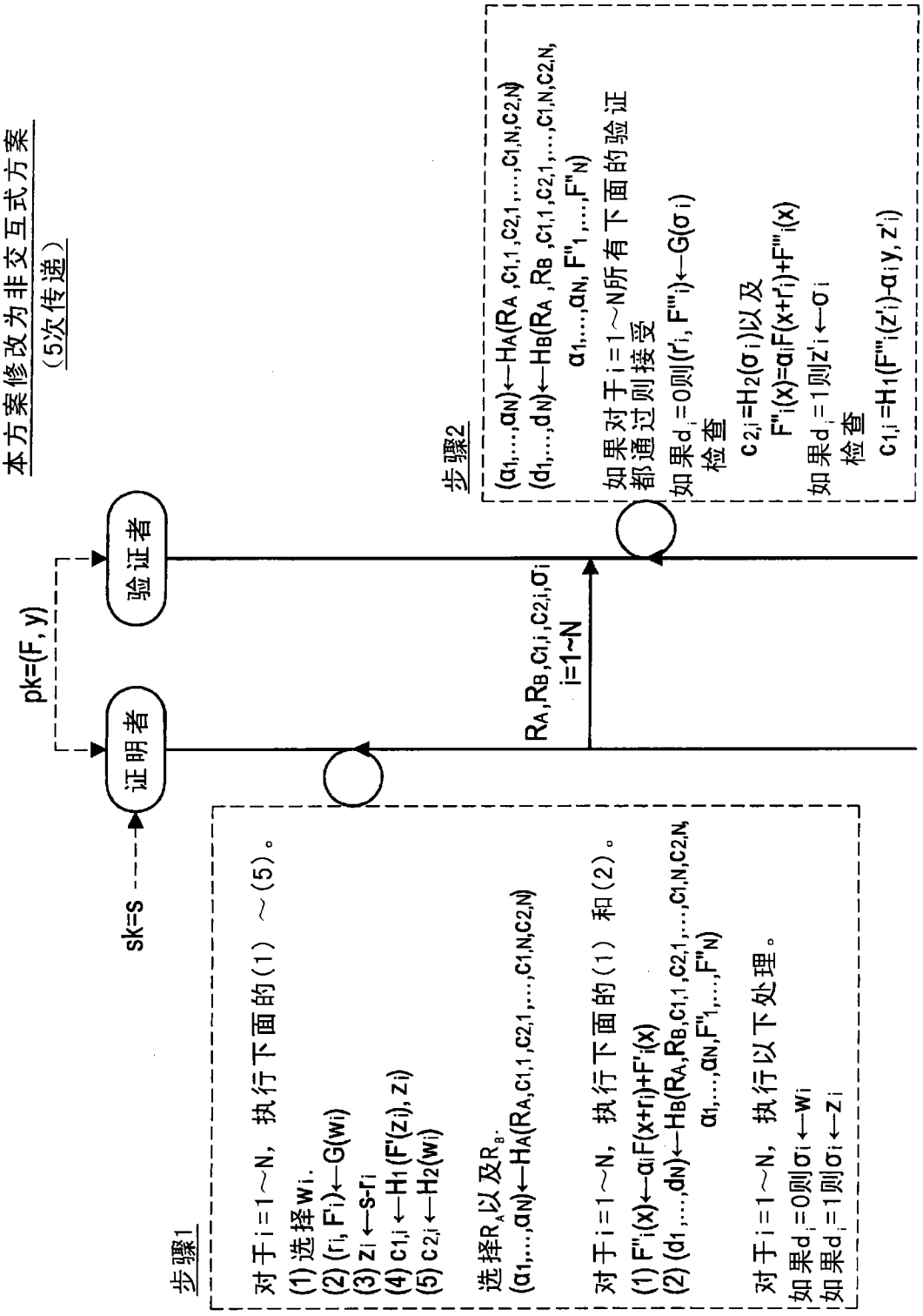

本方案修改为非交互式方案
（5次传递）
pk=(F, y)
sk=S
证明者
验证者
步骤1
对于i=1～N，执行下面的(1) ～(5)。
(1) 选择wi.
(2) (ri, F'i)←G(wi)
(3) zi ←s-ri
(4) c1,i ←H1(F'(zi), zi)
(5) c2,i ←H2(wi)
选择RA以及RB.
(α1,...,αN)←HA(RA,c1,1,c2,1,...,c1,N,c2,N)
对于i=1～N，执行下面的(1) 和(2)。
(1) F''i(x)←αiF(x+ri)+F'i(x)
(2) (d1,...,dN)←HB(RA,RB,c1,1,c2,1,...,c1,N,c2,N,
α1,...,αN,F''1,...,F''N)
对于i=1～N，执行以下处理。
如果di=0则σi ←wi
如果di=1则σi ←zi
RA,RB,c1,i,c2,i,σi
i=1~N
步骤2
(α1,...,αN)←HA(RA,c1,1,c2,1,...,c1,N,c2,N)
(d1,...,dN)←HB(RA,RB,c1,1,c2,1,...,c1,N,c2,N,
α1,...,αN, F''1,...,F''N)
如果对于i=1～N所有下面的验证
都通过则接受
如果di=0则(r'i, F''i)←G(σi)
检查
c2,i=H2(σi)以及
F''i(x)=αiF(x+r'i)+F''i(x)
如果di=1则z'i ←σi
检查
c1,i=H1(F''i(z'i)-αiy, z'i)

图16

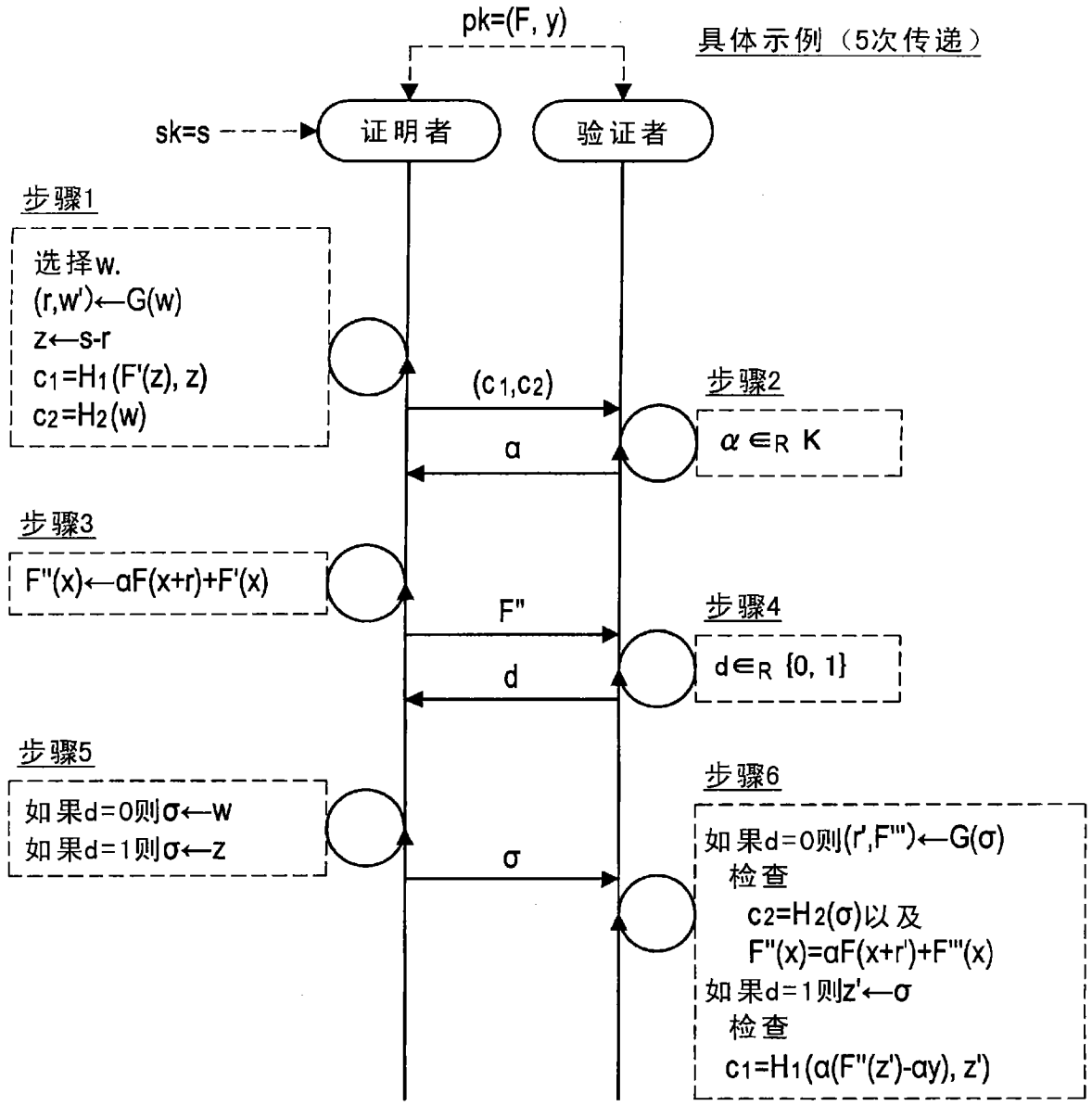

pk=(F, y)
具体示例（5次传递）
sk=s
证明者
验证者
步骤1
选择w.
(r,w')←G(w)
z←s-r
c1=H1(F'(z), z)
c2=H2(w)
(c1,c2)
步骤2
α ∈R K
α
步骤3
F''(x)←αF(x+r)+F'(x)
F''
步骤4
d∈R {0, 1}
d
步骤5
如果d=0则σ←w
如果d=1则σ←z
步骤6
σ
如果d=0则(r',F''')←G(σ)
检查
c2=H2(σ)以及
F''(x)=αF(x+r')+F'''(x)
如果d=1则z'←σ
检查
c1=H1(α(F''(z')-αy), z')

图17

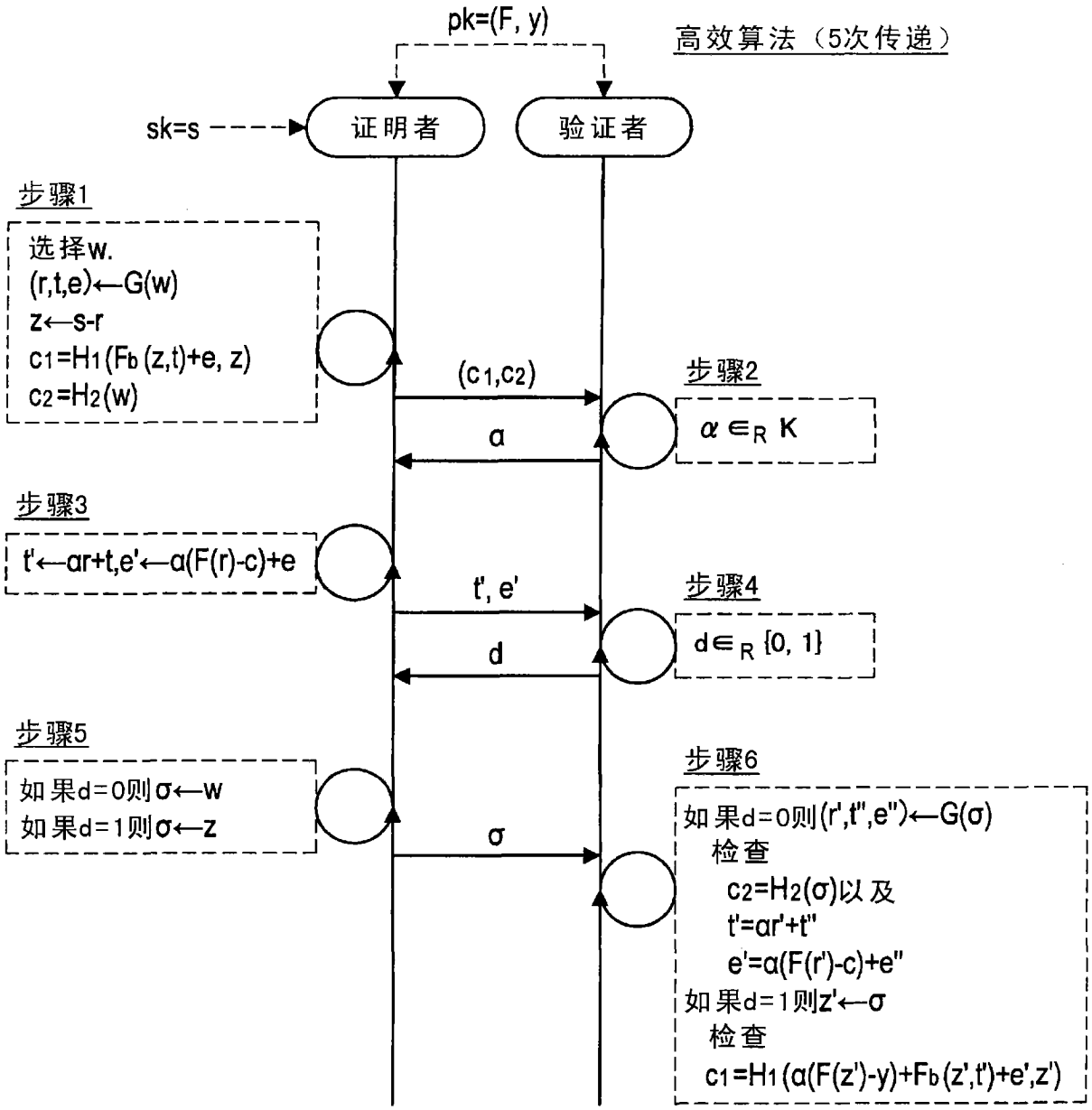

pk=(F, y)
高效算法（5次传递）
sk=s
证明者
验证者
步骤1
选择w.
(r,t,e)←G(w)
z←s-r
c1=H1(Fb(z,t)+e, z)
c2=H2(w)
(c1,c2)
步骤2
α ∈R K
α
步骤3
t'←αr+t,e'←α(F(r)-c)+e
t', e'
步骤4
d∈R {0, 1}
d
步骤5
如果d=0则σ←w
如果d=1则σ←z
σ
步骤6
如果d=0则(r',t'',e'')←G(σ)
检查
c2=H2(σ)以及
t'=αr'+t''
e'=α(F(r')-c)+e''
如果d=1则z'←σ
检查
c1=H1(α(F(z')-y)+Fb(z',t')+e',z')

图18

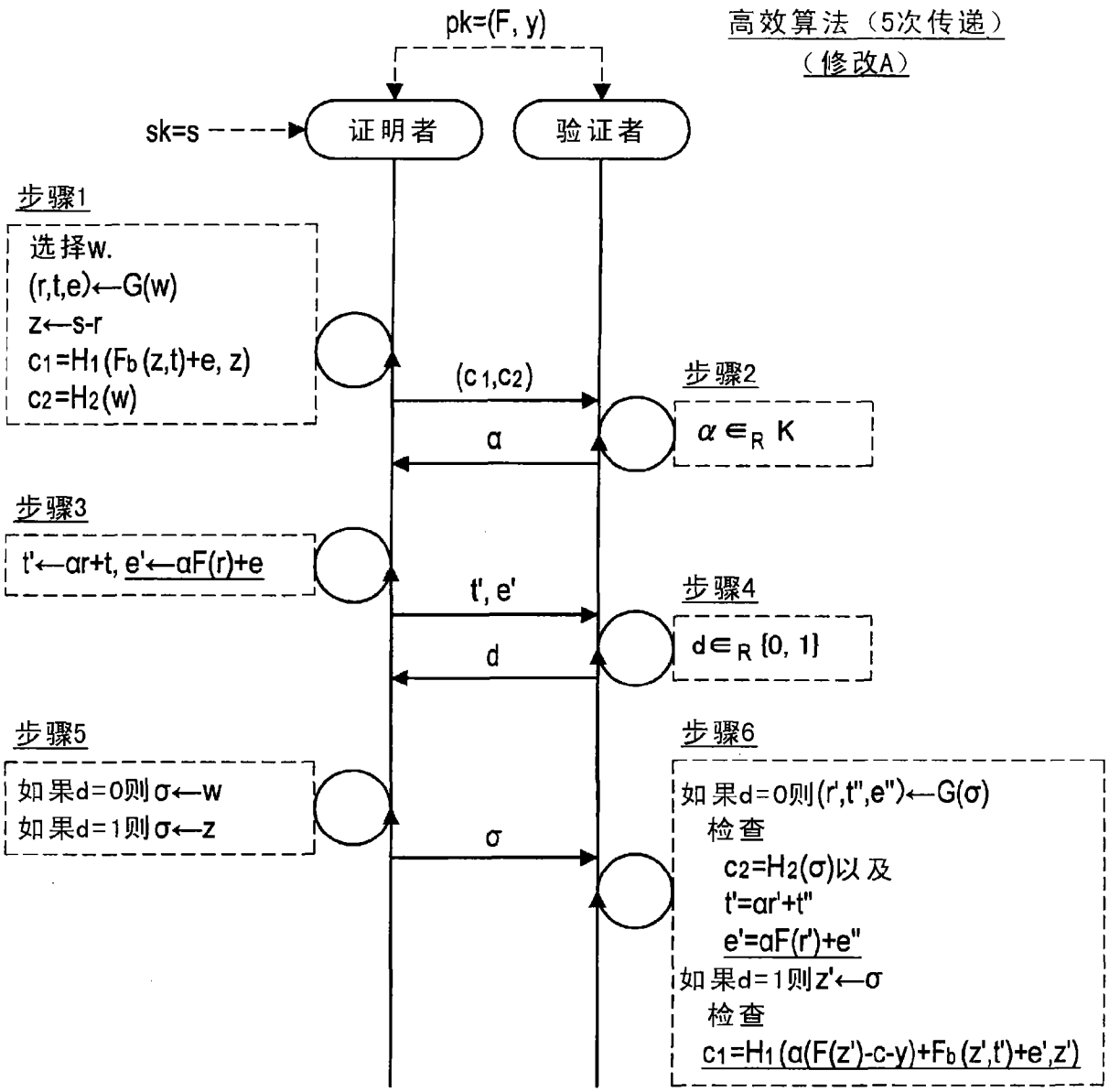

pk=(F, y)
高效算法（5次传递）
（修改A）
sk=s
证明者
验证者
步骤1
选择w.
(r,t,e)←G(w)
z←s-r
c1=H1(Fb(z,t)+e, z)
c2=H2(w)
(c1,c2)
步骤2
α ∈R K
α
步骤3
t'←αr+t, e'←αF(r)+e
t', e'
步骤4
d∈R {0, 1}
d
步骤5
如果d=0则σ←w
如果d=1则σ←z
σ
步骤6
如果d=0则(r',t",e")←G(σ)
检查
c2=H2(σ)以及
t'=αr'+t"
e'=αF(r')+e"
如果d=1则z'←σ
检查
c1=H1(α(F(z')-c-y)+Fb(z',t')+e',z')

图19

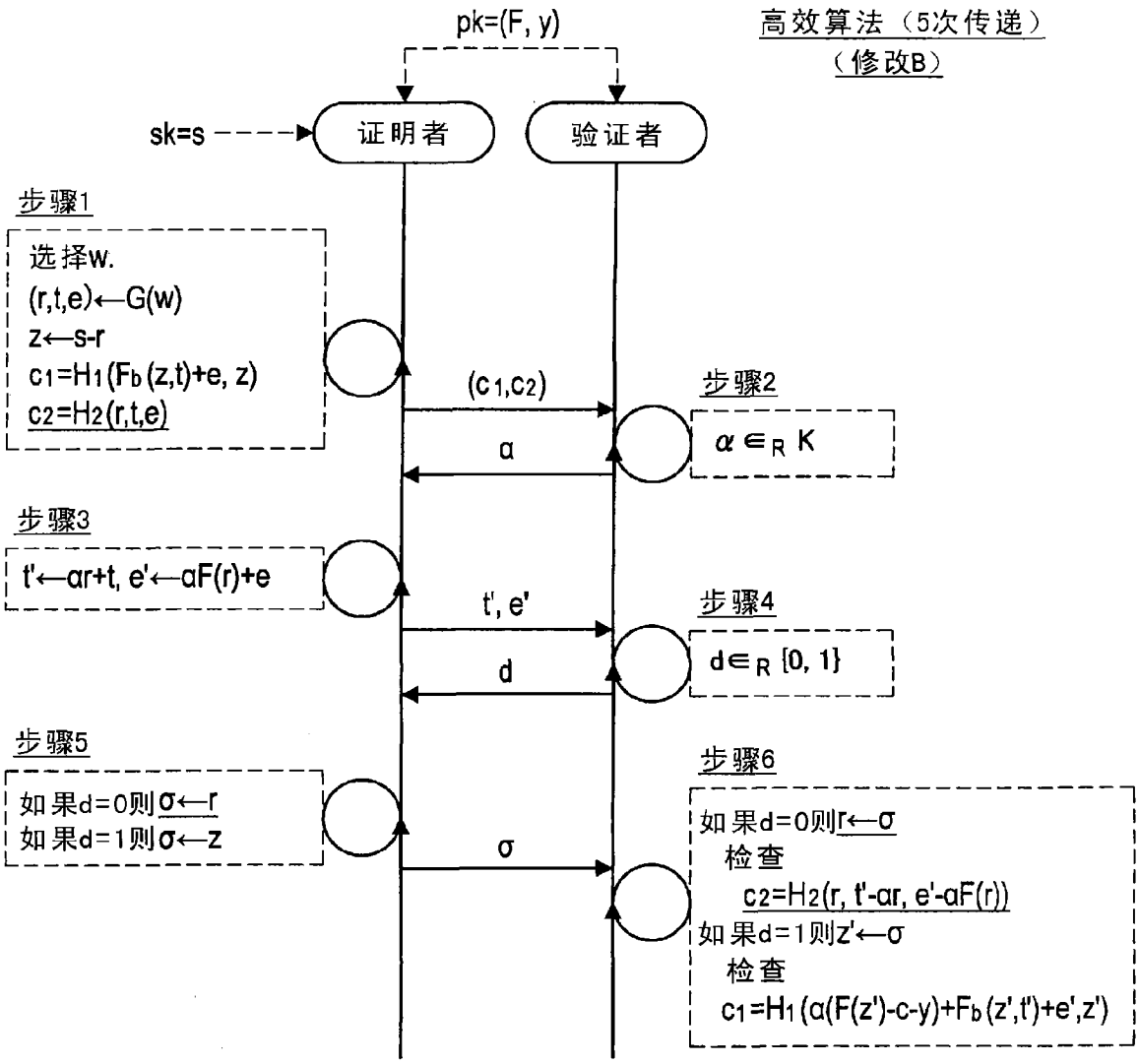

pk=(F, y)
高效算法（5次传递）
（修改B）
sk=s
证明者
验证者
步骤1
选择w.
(r,t,e)←G(w)
z←s-r
c1=H1(Fb(z,t)+e, z)
c2=H2(r,t,e)
(c1,c2)
步骤2
α
α∈R K
步骤3
t'←αr+t, e'←αF(r)+e
t', e'
步骤4
d
d∈R {0, 1}
步骤5
如果d=0则σ←r
如果d=1则σ←z
σ
步骤6
如果d=0则r←σ
检查
c2=H2(r, t'-αr, e'-αF(r))
如果d=1则z'←σ
检查
c1=H1(α(F(z')-c-y)+Fb(z',t')+e',z')

图20

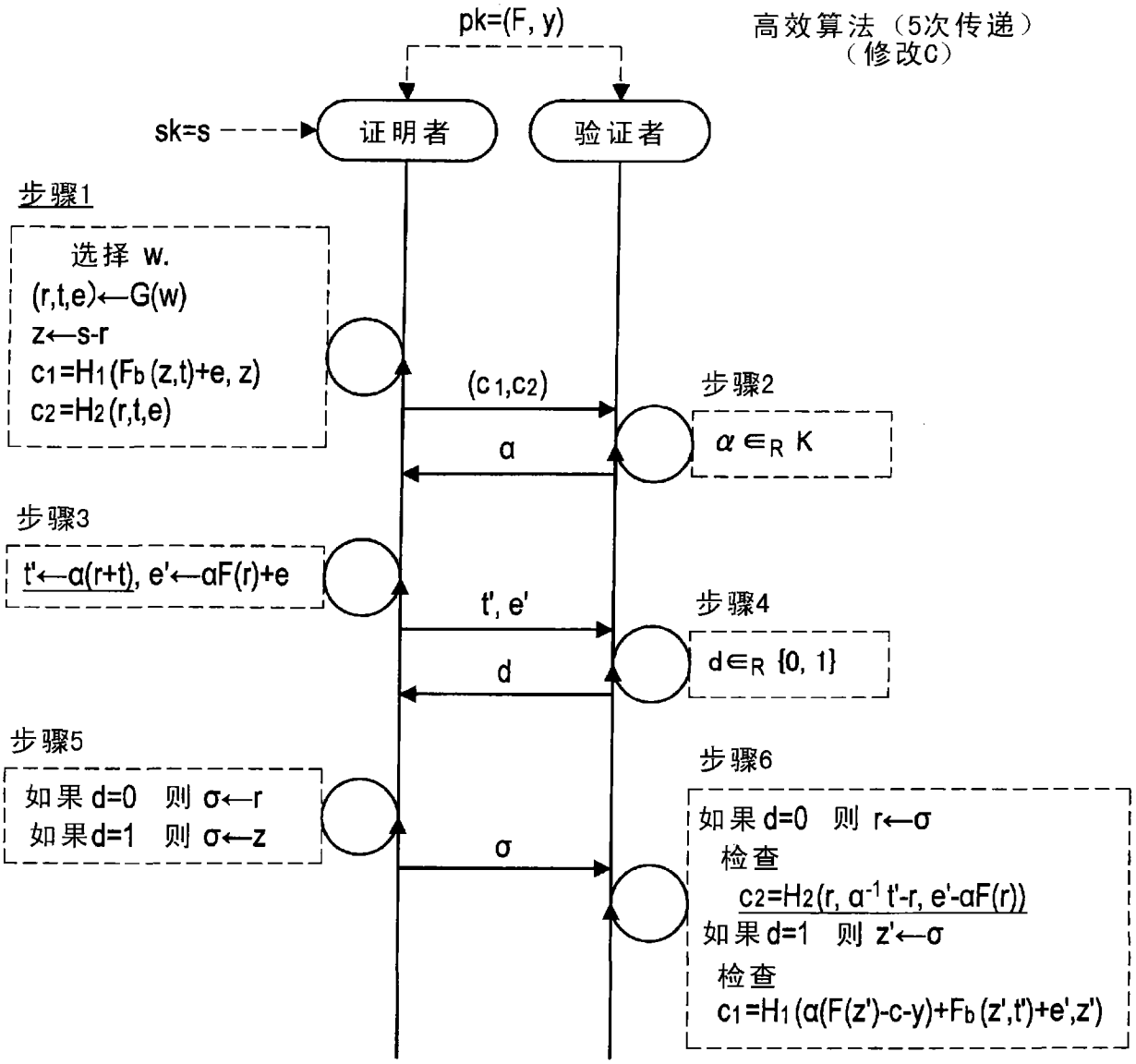

pk=(F, y)
高效算法（5次传递）
（修改c）
sk=s
证明者
验证者
步骤1
选择 w.
(r,t,e)←G(w)
z←s-r
c1=H1(Fb(z,t)+e, z)
c2=H2(r,t,e)
(c1,c2)
步骤2
α ∈R K
α
步骤3
t'←α(r+t), e'←αF(r)+e
t', e'
步骤4
d∈R {0, 1}
d
步骤5
如果 d=0 则 σ←r
如果d=1 则 σ←z
σ
步骤6
如果 d=0 则 r←σ
检查
c2=H2(r, α-1 t'-r, e'-αF(r))
如果 d=1 则 z'←σ
检查
c1=H1(α(F(z')-c-y)+Fb(z',t')+e',z')

图21

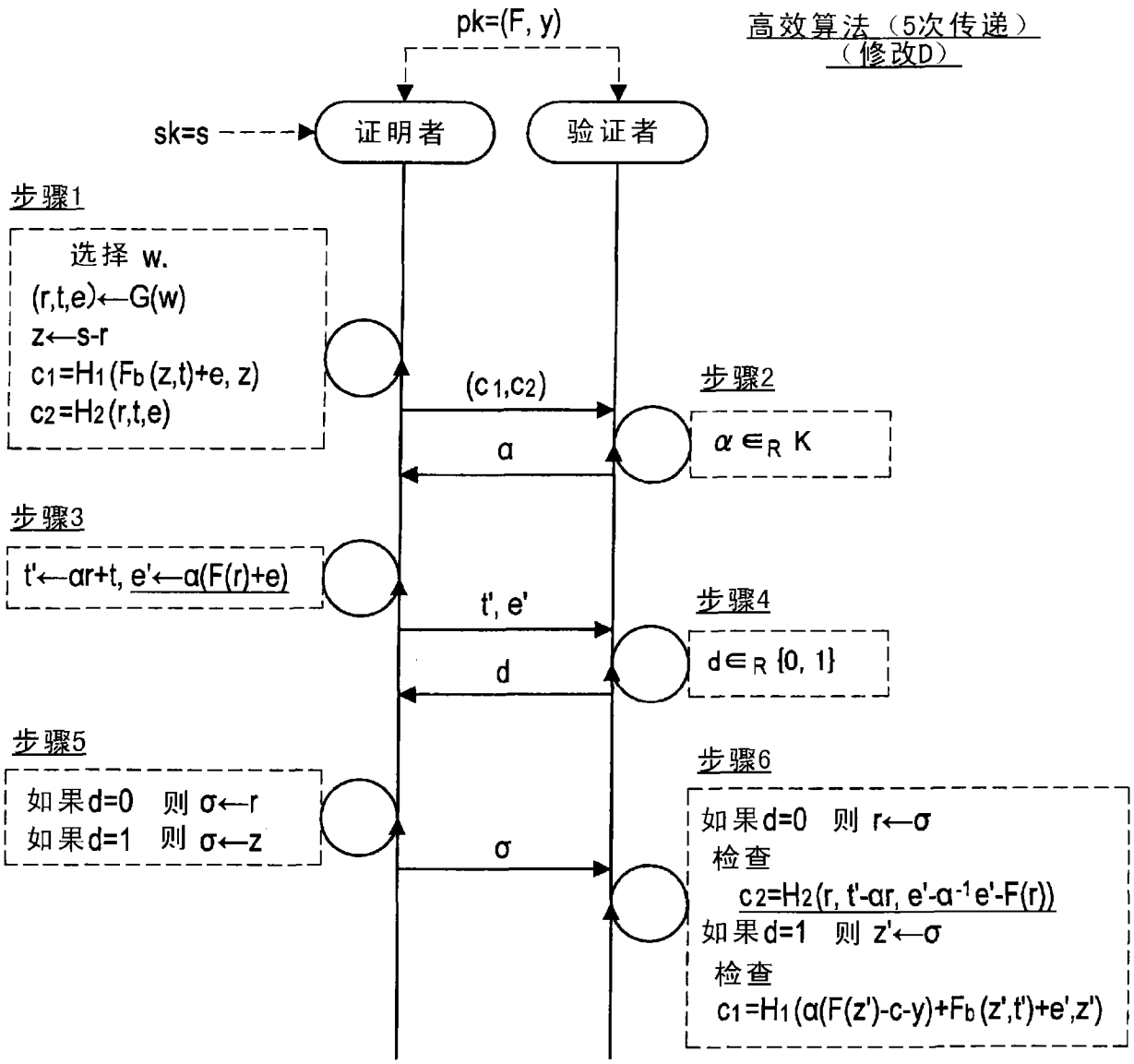

pk=(F, y)
高效算法（5次传递）
（修改D）
sk=s
证明者
验证者
步骤1
选择 w.
(r,t,e)←G(w)
z←s-r
c1=H1(Fb(z,t)+e, z)
c2=H2(r,t,e)
(c1,c2)
步骤2
α ∈R K
α
步骤3
t'←αr+t, e'←α(F(r)+e)
t', e'
步骤4
d∈R {0, 1}
d
步骤5
如果d=0 则 σ←r
如果d=1 则 σ←z
σ
步骤6
如果d=0 则 r←σ
检查
c2=H2(r, t'-αr, e'-α-1e'-F(r))
如果d=1 则 z'←σ
检查
c1=H1(α(F(z')-c-y)+Fb(z',t')+e',z')

图22

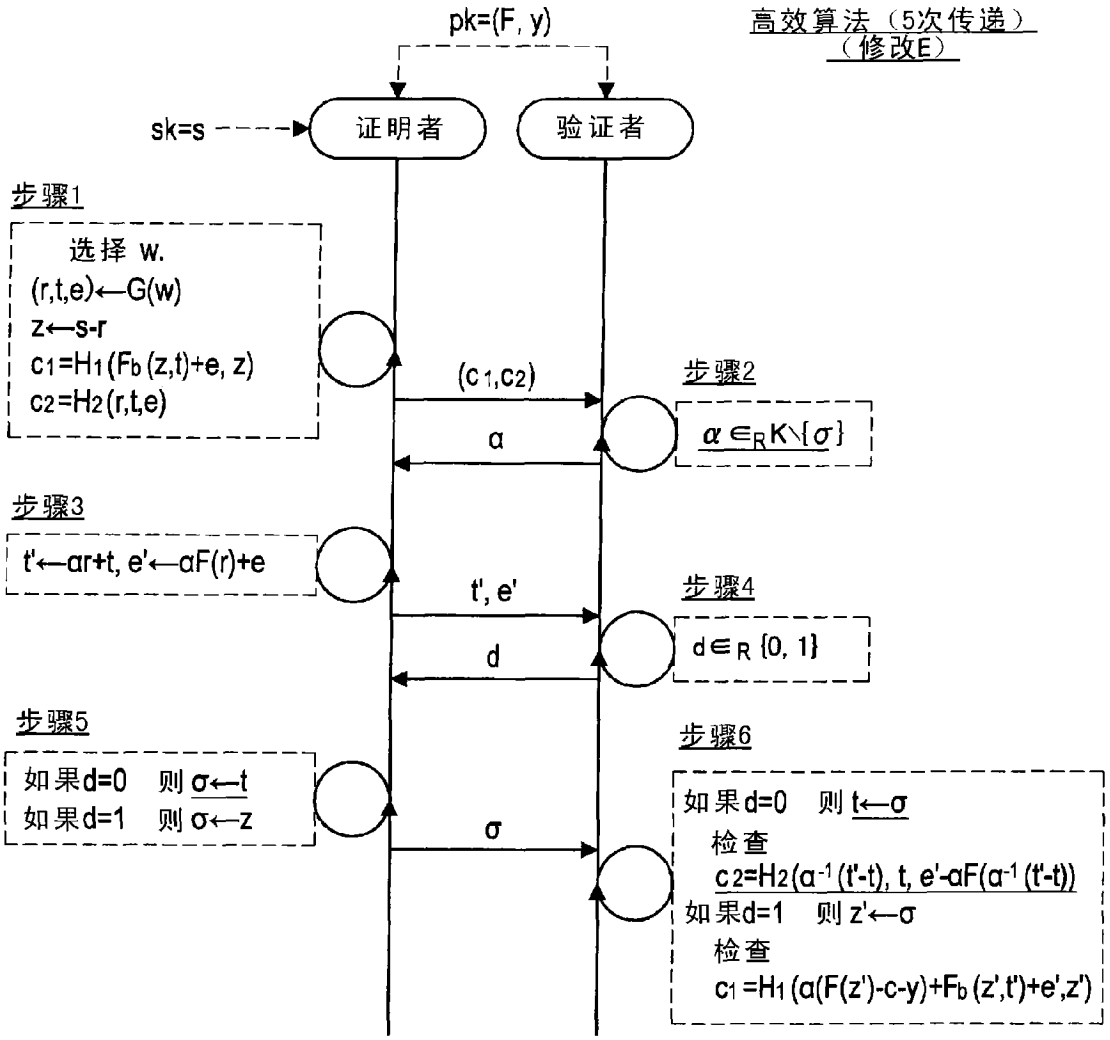

pk=(F, y)
高效算法（5次传递）
（修改E）
sk=s
证明者
验证者
步骤1
选择 w.
(r,t,e)←G(w)
z←s-r
c1=H1(Fb(z,t)+e, z)
c2=H2(r,t,e)
(c1,c2)
步骤2
α∈RK\{σ}
α
步骤3
t'←αr+t, e'←αF(r)+e
t', e'
步骤4
d∈R {0, 1}
d
步骤5
如果d=0 则 σ←t
如果d=1 则 σ←z
σ
步骤6
如果d=0 则 t←σ
检查
c2=H2(α-1(t'-t), t, e'-αF(α-1(t'-t))
如果d=1 则 z'←σ
检查
c1=H1(α(F(z')-c-y)+Fb(z',t')+e',z')

图23

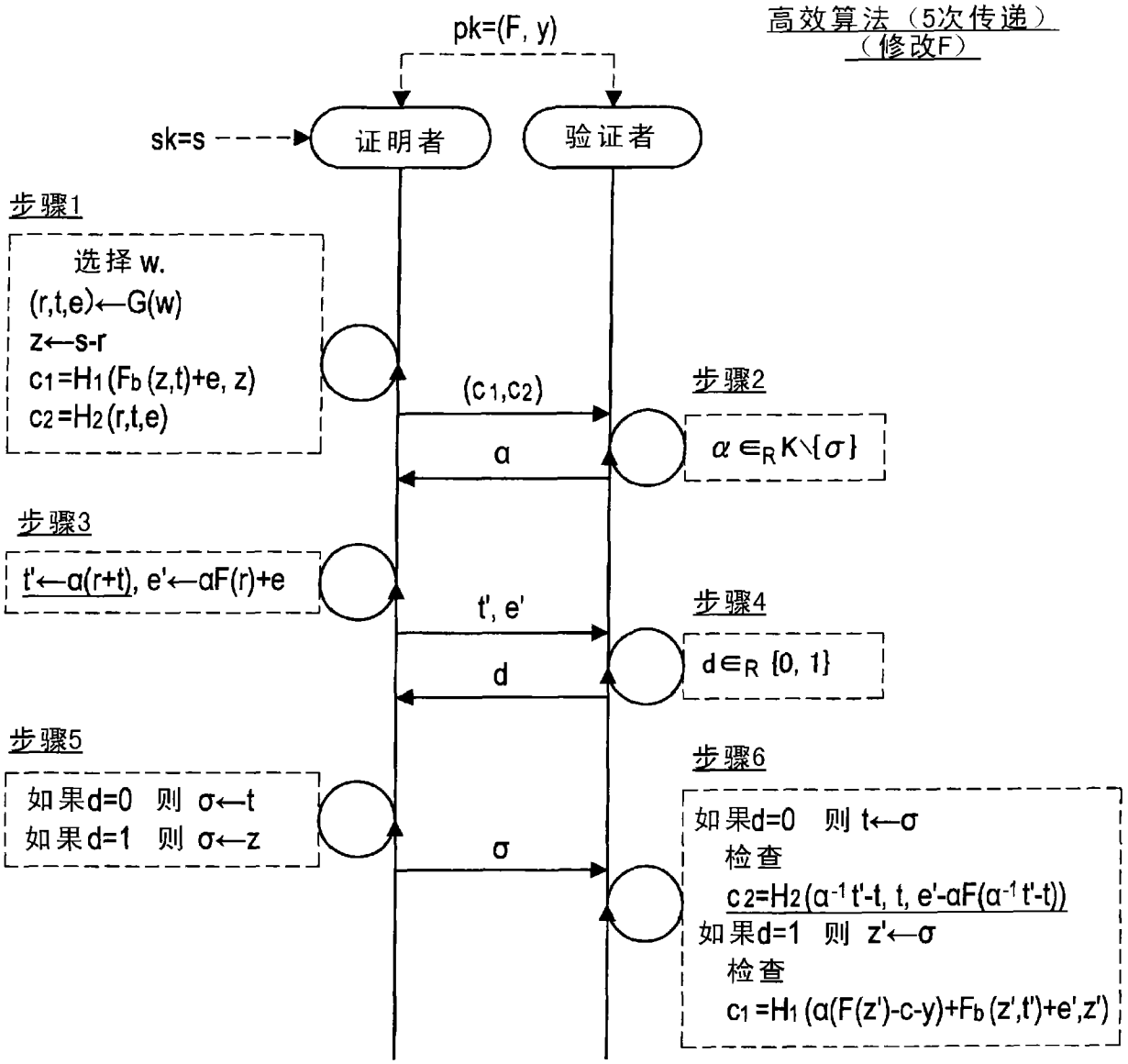

高效算法（5次传递）
（修改F）
pk=(F, y)
sk=s
证明者
验证者
步骤1
选择 w.
(r,t,e)←G(w)
z←s-r
c1=H1(Fb(z,t)+e, z)
c2=H2(r,t,e)
(c1,c2)
步骤2
α ∈R K\{σ}
α
步骤3
t'←α(r+t), e'←αF(r)+e
t', e'
步骤4
d∈R {0, 1}
d
步骤5
如果d=0 则 σ←t
如果d=1 则 σ←z
σ
步骤6
如果d=0 则 t←σ
检查
c2=H2(α-1 t'-t, t, e'-αF(α-1 t'-t))
如果d=1 则 z'←σ
检查
c1=H1(α(F(z')-c-y)+Fb(z',t')+e',z')

图24

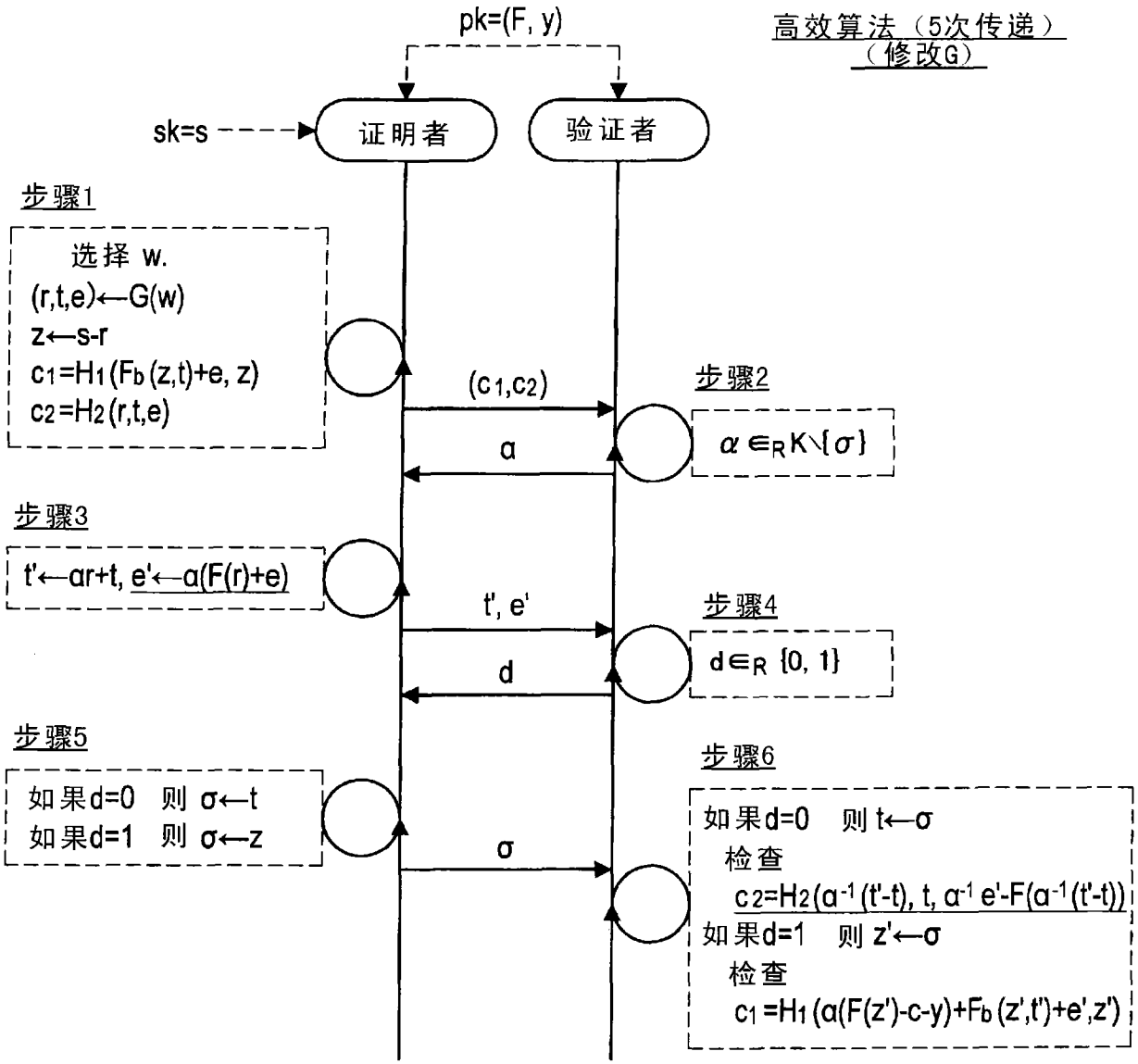

pk=(F, y)
高效算法（5次传递）
（修改G）
sk=s
证明者
验证者
步骤1
选择 w.
(r,t,e)←G(w)
z←s-r
c1=H1(Fb(z,t)+e, z)
c2=H2(r,t,e)
(c1,c2)
步骤2
α ∈R K\{σ}
α
步骤3
t'←αr+t, e'←α(F(r)+e)
t', e'
步骤4
d∈R {0, 1}
d
步骤5
如果d=0 则 σ←t
如果d=1 则 σ←z
σ
步骤6
如果d=0 则 t←σ
检查
c2=H2(α-1(t'-t), t, α-1 e'-F(α-1(t'-t))
如果d=1 则 z'←σ
检查
c1=H1(α(F(z')-c-y)+Fb(z',t')+e',z')

图25

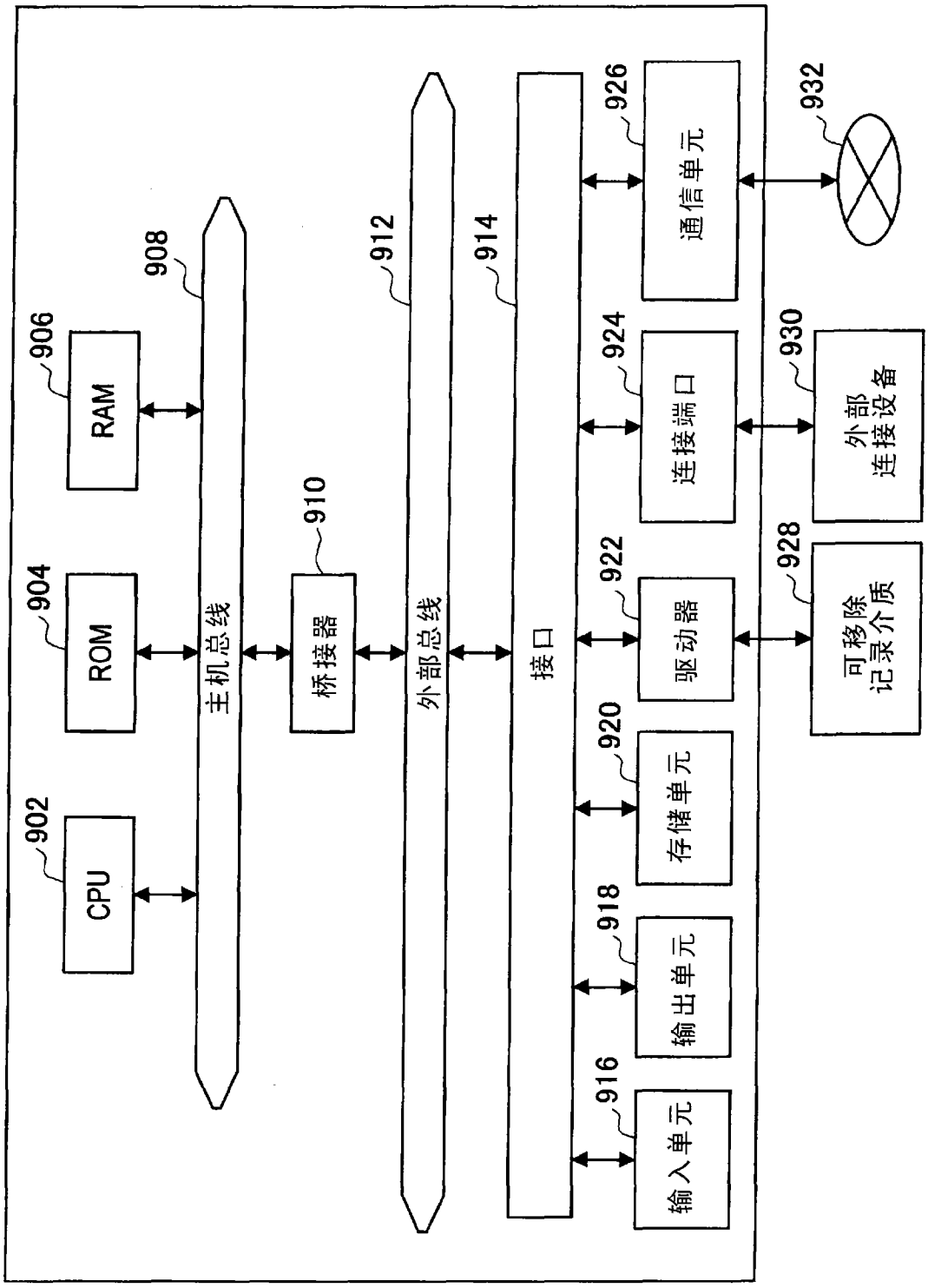

902
CPU
904
ROM
906
RAM
908
主机总线
910
桥接器
912
外部总线
914
接口
916
输入单元
918
输出单元
920
存储单元
922
驱动器
924
连接端口
926
通信单元
928
可移除记录介质
930
外部连接设备
932

图26

表：3次传递与5次传递之间的效率比较

	3次传递	5次传递				
环的元素数目q	—	$q=2^4$	$q=2^5$	$q=2^6$	$q=2^7$	$q=2^8$
每次交互式协议的伪造成功概率	0.667	0.531	0.516	0.508	0.504	0.502
n比特的安全性所需的重复次数	1.710n	1.096n	1.046n	1.023n	1.011n	1.006n
80比特的安全性所需的重复次数	137	88	84	82	81	81
128比特的安全性所需的重复次数	219	141	134	131	130	129

图27

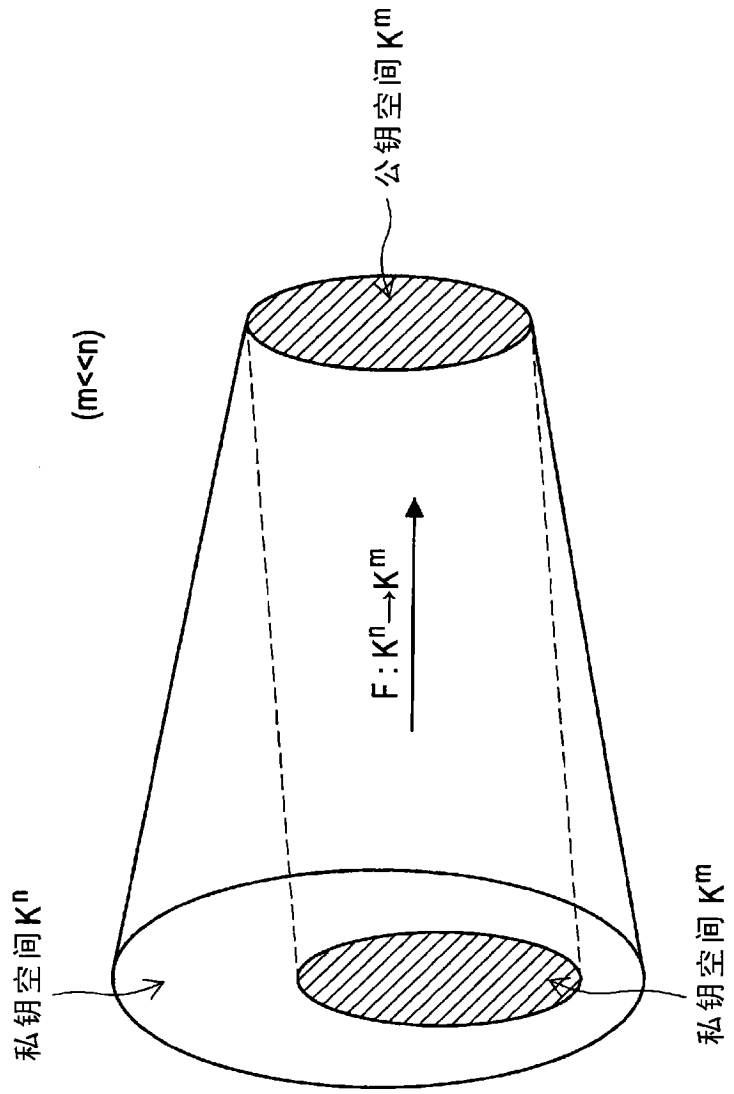


图28