(19)中華民國智慧財產局
(12)發明說明書公開本 (11)公開編號：TW 201320701 A1
(43)公開日： 中華民國 102 (2013) 年 05 月 16 日

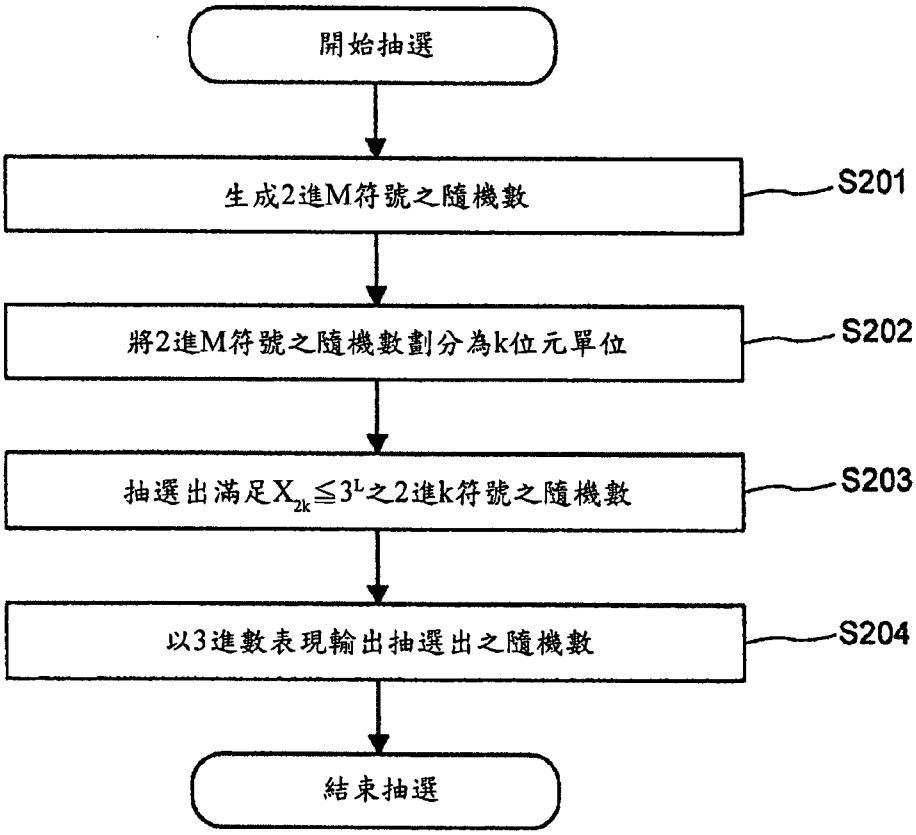
(21)申請案號：101128119 (22)申請日： 中華民國 101 (2012) 年 08 月 03 日
(51)Int. Cl.： *H04L9/28 (2006.01)* *H04L9/32 (2006.01)*
G09C1/00 (2006.01)
(30)優先權：2011/08/29 日本 2011-185947
2012/03/02 日本 2012-046687
(71)申請人：新力股份有限公司 (日本) SONY CORPORATION (JP)
日本
(72)發明人：作本紘一 SAKUMOTO, KOICHI (JP) ；白井太三 SHIRAI, TAIZO (JP) ；桶渡玄良 HIWATARI, HARUNAGA (JP) ；神尾一也 KAMIO, KAZUYA (JP)
(74)代理人：陳長文
申請實體審查：無 申請專利範圍項數：19 項 圖式數：28 共 129 頁

(54)名稱

資訊處理裝置、資訊處理方法及程式

(57)摘要

本發明之目的在於可高效地代入多次多元多項式之係數。本發明提供一種資訊處理裝置，其包括：數生成部，其根據在執行利用包含多次多元多項式之組 F=(f_1，...，f_m)之公鑰的公鑰認證方式或電子署名方式之演算法的實體間所共有之資訊，使用特定之函數，而生成構成上述多次多元多項式之組 F 之各項之係數中所使用之數；及分配部，其將上述數生成部生成之數分配至構成要素中包含上述多次多元多項式之組 F 的多次多元多項式之係數；且上述分配部將構成要素中包含上述多次多元多項式之組 F 的多次多元多項式之係數中、變數之組合之種類為相同項之係數群組化，而以群組單位執行分配處理。


開始抽選
生成2進M符號之隨機數
S201
將2進M符號之隨機數劃分為k位元單位
S202
抽選出滿足$X_{2k} \leqq 3^L$之2進k符號之隨機數
S203
以3進數表現輸出抽選出之隨機數
S204
結束抽選

(19)中華民國智慧財產局
(12)發明說明書公開本 (11)公開編號：TW 201320701 A1
(43)公開日： 中華民國 102 (2013) 年 05 月 16 日

(21)申請案號：101128119 (22)申請日： 中華民國 101 (2012) 年 08 月 03 日
(51)Int. Cl.： *H04L9/28 (2006.01)* *H04L9/32 (2006.01)*
G09C1/00 (2006.01)
(30)優先權：2011/08/29 日本 2011-185947
2012/03/02 日本 2012-046687
(71)申請人：新力股份有限公司 (日本) SONY CORPORATION (JP)
日本
(72)發明人：作本紘一 SAKUMOTO, KOICHI (JP) ；白井太三 SHIRAI, TAIZO (JP) ；桶渡玄良 HIWATARI, HARUNAGA (JP) ；神尾一也 KAMIO, KAZUYA (JP)
(74)代理人：陳長文
申請實體審查：無 申請專利範圍項數：19 項 圖式數：28 共 129 頁

(54)名稱

資訊處理裝置、資訊處理方法及程式

(57)摘要

本發明之目的在於可高效地代入多次多元多項式之係數。本發明提供一種資訊處理裝置，其包括：數生成部，其根據在執行利用包含多次多元多項式之組 F=(f_1，...，f_m)之公鑰的公鑰認證方式或電子署名方式之演算法的實體間所共有之資訊，使用特定之函數，而生成構成上述多次多元多項式之組 F 之各項之係數中所使用之數；及分配部，其將上述數生成部生成之數分配至構成要素中包含上述多次多元多項式之組 F 的多次多元多項式之係數；且上述分配部將構成要素中包含上述多次多元多項式之組 F 的多次多元多項式之係數中、變數之組合之種類為相同項之係數群組化，而以群組單位執行分配處理。

發明專利說明書

（本說明書格式、順序及粗體字，請勿任意更動，※記號部分請勿填寫）

※ 申請案號：101128119

※ 申請日：101.8.3　　※IPC 分類：H04L 9/08 (2006.01)
H04L 9/32 (2006.01)
G09C 1/00 (2006.01)

一、發明名稱：（中文/英文）

資訊處理裝置、資訊處理方法及程式

二、中文發明摘要：

本發明之目的在於可高效地代入多次多元多項式之係數。本發明提供一種資訊處理裝置，其包括：數生成部，其根據在執行利用包含多次多元多項式之組$F=(f_1, \ldots, f_m)$之公鑰的公鑰認證方式或電子署名方式之演算法的實體間所共有之資訊，使用特定之函數，而生成構成上述多次多元多項式之組F之各項之係數中所使用之數；及分配部，其將上述數生成部生成之數分配至構成要素中包含上述多次多元多項式之組F的多次多元多項式之係數；且上述分配部將構成要素中包含上述多次多元多項式之組F的多次多元多項式之係數中、變數之組合之種類為相同項之係數群組化，而以群組單位執行分配處理。

三、英文發明摘要：

四、指定代表圖：

(一)本案指定代表圖為：第（20）圖。

(二)本代表圖之元件符號簡單說明：

（無元件符號說明）

五、本案若有化學式時，請揭示最能顯示發明特徵的化學式：

（無）

六、發明說明：

【發明所屬之技術領域】

本技術係關於一種資訊處理裝置、資訊處理方法及程式。

【先前技術】

隨著資訊處理技術或通訊技術之快速之發展，不論公務文件或是私人文件，文件之電子化均急速地發展。隨之，多數個人或企業對電子文件之安全管理寄予較大之關心。受此種關心之增強之影響，而盛行研究於各方面針對電子文件之監聽或偽造等篡改(Tampering)行為的對策。針對電子文件之竊聽，例如藉由對電子文件加密，而確保安全性。又，針對電子文件之偽造，例如藉由利用電子署名，而確保安全性。但，只要利用之密碼或電子署名未具有較高之耐干擾性，便無法保證充分之安全性。

電子署名係用以特定電子文件之製作者。因此，電子署名應僅可由電子文件之製作者生成。假如存在惡意之第三者可生成相同之電子署名，則該第三者可冒充成電子文件之製作者。即，存在惡意之第三者會偽造電子文件。為防止此種偽造，對電子署名之安全性進行了各種討論。作為目前廣泛利用之電子署名方式，例如已知RSA署名方式或DSA署名方式等。

RSA署名方式係以「對較大之合成數進行素因數分解之困難性(以下稱為素因數分解問題)」為安全性之依據。又，DSA署名方式係以「對離散對數問題求解之導出之困

難性」為安全性之依據。該等依據之原因在於不存在利用古典之電腦高效地解開素因數分解問題或離散對數問題之演算法。即，上述困難性意味著古典之電腦中之計算量之困難性。然而，可以說若使用量子電腦(quantum computer)，則會高效地算出對素因數分解問題或離散對數問題的解答。

目前利用之電子署名方式或公鑰認證方式多與RSA署名方式或DSA署名方式同樣地以素因數分解問題或離散對數問題之困難性為安全性之依據。因此，此種電子署名方式或公鑰認證方式於量子電腦得到實用化之情形時，無法確保其安全性。因此，需要實現以與素因數分解問題或離散對數問題等由量子電腦容易地解開之問題不同之問題為安全性之依據的新電子署名方式及公鑰認證方式。作為不易由量子電腦容易地解開之問題，例如有多元多項式問題。

作為以多元多項式問題為安全性之依據之電子署名方式，例如已知基於MI(Matsumoto-Imaicryptography，松本-今井密碼法)、HFE(Hidden Field Equation cryptography，隱藏欄位方程式密碼法)、OV(Oil-Vinegar signature scheme，Oil-Vinegar簽章機制)、TTM(Tamed Transformation Method cryptography，馴轉換密碼法)的方式。例如於下述非專利文獻1、2中揭示有基於HFE之電子署名方式。

[先前技術文獻]

[非專利文獻]

[非專利文獻1] Jacques Patarin Asymmetric Cryptography

with a Hidden Monomial. CRYPTO 1996, pp. 45-60。

[非專利文獻2] Patarin, J., Courtois, N., and Goubin, L. 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。

【發明內容】

[發明所欲解決之問題]

如上所述，多元多項式問題係即便利用量子電腦亦不易解開之稱為NP(Non-Deterministic Polynomial，非確定性多項式)困難問題的問題之一例。通常，利用以HFE等為代表之多元多項式問題的公鑰認證方式係利用置入有特殊之陷門(Trap Door)的多次多元聯立方程式。例如準備關於x_1，...，x_n之多次多元聯立方程式$F(x_1, ..., x_n)=y$以及線性轉換A及B，秘密地管理線性轉換A及B。於該情形時，多次多元聯立方程式F、線性轉換A及B成為陷門。

知曉陷門F、A、B之實體(entity)可解開關於x_1，...，x_n之方程式$B(F(A(x_1, ..., x_n)))=y'$。另一方面，未知曉陷門F、A、B之實體無法解開關於x_1，...，x_n之方程式$B(F(A(x_1, ..., x_n)))=y'$。藉由利用該結構，而實現以多次多元聯立方程式之解答困難性為安全性之依據的公鑰認證方式或電子署名方式。

如上所述，為實現此種公鑰認證方式或電子署名方式，必需準備如滿足$B(F(A(x_1, ..., x_n)))=y$般之特殊之多次多

元聯立方程式。又，於生成署名時，必需解開多次多元聯立方程式F。因此，可利用之多次多元聯立方程式F僅限於相對容易解開者。即，於至此為止之方式中，僅可使用合成相對容易解開之3個函數(陷門)B、F、A之形式的多次多元聯立方程式$B(F(A(x_1, \ldots, x_n)))=y$，而難以確保充分之安全性。

因此，本件發明者鑒於上述實際情況，利用高效地解開之方法(陷門)未知之多次多元聯立方程式，創作出具有較高之安全性之高效之公鑰認證方式及電子署名方式(Koichi Sakumoto, Taizo Shirai and Harunaga Hiwatari,「Public-Key Identification Schemes Based on Multivariate Quadratic Polynomials」, CRYPTO 2011, LNCS 6841, pp. 706-723, 2011.)。

然而，於該公鑰認證方式及電子署名方式中利用之多元多項式之係數的數量較多，為了用作公鑰，必需預先決定於證明者與驗證者之間高效地分配係數之方法。本技術係以可高效地代入多元多項式之係數的新穎且經改良之資訊處理裝置、資訊處理方法及程式之提供為意圖創作而成者。

[解決問題之技術手段]

根據本技術之某觀點，提供一種資訊處理裝置，其包括：數生成部，其係根據在執行利用包含多次多元多項式之組$F=(f_1, \ldots, f_m)$之公鑰的公鑰認證方式或電子署名方式之演算法的實體間所共有之資訊，使用特定之函數，而

生成構成上述多次多元多項式之組F之各項之係數中所使用之數；及分配部，其將上述數生成部生成之數分配至構成要素中包含上述多次多元多項式之組F的多次多元多項式之係數；且上述分配部將構成要素中包含上述多次多元多項式之組F的多次多元多項式之係數中、變數之組合之種類為相同項之係數群組化，而以群組單位執行分配處理。

又，根據本技術之其他觀點，提供一種資訊處理裝置，其包括：數生成部，其根據在執行利用包含多次多元多項式之組$F=(f_1, \ldots, f_m)$之公鑰的公鑰認證方式或電子署名方式之演算法的實體間所共有之資訊，使用特定之函數，而生成構成上述多次多元多項式之組F之各項之係數中所使用之數；及分配部，其將上述數生成部生成之數分配至構成要素中包含上述多次多元多項式之組F的多次多元多項式之係數；且上述分配部將構成要素中包含上述多次多元多項式之組F的多次多元多項式之以2次式(Quadratic Form)表現之情形時的係數矩陣以每列或每行進行群組化，而以群組單位執行分配處理。

又，根據本技術之其他觀點，提供一種資訊處理裝置，其包括：數生成部，其根據在執行利用包含多次多元多項式之組$F=(f_1, \ldots, f_m)$之公鑰的公鑰認證方式或電子署名方式之演算法的實體間所共有之資訊，使用特定之函數，而生成與構成上述多次多元多項式之組F的各項之係數相同數量之數；及分配部，其將按照在上述證明者與上述認

證者之間預先決定之順序而由上述數生成部生成之數，分配至構成要素中包含上述多次多元多項式之組F的多次多元多項式之係數。

又，根據本技術之其他觀點，提供一種資訊處理方法，其包括以下步驟：根據在執行利用包含多次多元多項式之組$F=(f_1, \ldots, f_m)$之公鑰的公鑰認證方式或電子署名方式之演算法的實體間所共有之資訊，使用特定之函數，而生成構成上述多次多元多項式之組F之各項之係數中所使用之數；及將所生成之數分配至構成要素中包含上述多次多元多項式之組F的多次多元多項式之係數；且於上述分配步驟中，將構成要素中包含上述多次多元多項式之組F的多次多元多項式之係數中、變數之組合之種類為相同項之係數群組化，而以群組單位執行分配處理。

又，根據本技術之其他觀點，提供一種資訊處理方法，其包括以下步驟：根據在執行利用包含多次多元多項式之組$F=(f_1, \ldots, f_m)$之公鑰的公鑰認證方式或電子署名方式之演算法的實體間所共有之資訊，使用特定之函數，而生成構成上述多次多元多項式之組F之各項之係數中所使用之數；及將所生成之數分配至構成要素中包含上述多次多元多項式之組F的多多元多項式之係數；且於上述分配步驟中，將構成要素中包含上述多次多元多項式之組F的多次多元多項式之以2次式表現之情形時的係數矩陣以每列或每行進行群組化，而以群組單位執行分配處理。

又，根據本技術之其他觀點，提供一種資訊處理方法，

其包括以下步驟：根據在執行利用包含多次多元多項式之組$F=(f_1, \ldots, f_m)$之公鑰的公鑰認證方式或電子署名方式之演算法的實體間所共有之資訊，使用特定之函數，而生成與構成上述多次多元多項式之組F的各項之係數相同數量之數；及將按照在上述證明者與上述認證者之間預先決定之順序而由上述數生成部生成之數，分配至構成要素中包含上述多次多元多項式之組F的多次多元多項式之係數。

又，根據本技術之其他觀點，提供一種程式，其係用以使電腦實現以下功能者：數生成功能，其根據在執行利用包含多次多元多項式之組$F=(f_1, \ldots, f_m)$之公鑰的公鑰認證方式或電子署名方式之演算法的實體間所共有之資訊，使用特定之函數，而生成構成上述多次多元多項式之組F之各項之係數中所使用之數；及分配功能，其係將上述數生成功能所生成之數分配至構成要素中包含上述多次多元多項式之組F的多次多元多項式之係數；且上述分配功能將構成要素中包含上述多次多元多項式之組F的多次多元多項式之係數中、變數之組合之種類為相同項之係數群組化，而以群組單位執行分配處理。

又，根據本技術之其他觀點，提供一種程式，其係用以使電腦實現以下功能者：數生成功能，其根據在執行利用包含多次多元多項式之組$F=(f_1, \ldots, f_m)$之公鑰的公鑰認證方式或電子署名方式之演算法的實體間所共有之資訊，使用特定之函數，而生成構成上述多次多元多項式之組F

之各項之係數中所使用之數；及分配功能，其係將上述數生成功能所生成之數分配至構成要素中包含上述多次多元多項式之組F的多次多元多項式之係數；且上述分配功能將構成要素中包含上述多次多元多項式之組F的多次多元多項式之以2次式表現之情形時的係數矩陣以每列或每行進行群組化，而以群組單位執行分配處理。

又，根據本技術之其他觀點，提供一種程式，其係用以使電腦實現以下功能者：數生成功能，其根據在執行利用包含多次多元多項式之組$F=(f_1, \ldots, f_m)$之公鑰的公鑰認證方式或電子署名方式之演算法的實體間所共有之資訊，使用特定之函數，而生成與構成上述多次多元多項式之組F的各項之係數相同數量之數；及分配功能，其將按照在上述證明者與上述認證者之間預先決定之順序而由上述數生成部生成之數，分配至構成要素中包含上述多次多元多項式之組F的多次多元多項式之係數。

又，根據本技術之其他觀點，提供一種記錄媒體，其記錄有上述程式，且可由電腦讀取。

[發明之效果]

如以上所說明般，根據本技術，可高效地代入多元多項式之係數。

【實施方式】

以下，一面參照隨附圖式一面對本技術之較佳之實施形態詳細地進行說明。再者，於本說明書及圖式中，對實質上具有同一功能構成之構成要素標註同一符號，藉此省略

重複說明。

[關於說明之流程]

此處，對與以下記載之本技術之實施形態相關之說明之流程簡單地進行敘述。首先，一面參照圖1一面對公鑰認證方式之演算法構成進行說明。其次，一面參照圖2一面對電子署名方式之演算法構成進行說明。其次，一面參照圖3一面對n路徑之公鑰認證方式進行說明。

其次，一面參照圖4~圖6一面對3路徑之公鑰認證方式之演算法之構成例進行說明。其次，一面參照圖7~圖9一面對5路徑之公鑰認證方式之演算法之構成例進行說明。其次，一面參照圖10及圖11一面對將3路徑及5路徑之公鑰認證方式之高效之演算法變形成電子署名方式之演算法之方法進行說明。

其次，一面參照圖12~圖16一面對當執行本實施形態之電子署名方式之演算法時削減署名驗證所需之記憶體量之方法進行說明。其次，一面參照圖17~圖21一面對自2進數隨機數高效地抽選出3進數隨機數之方法進行說明。其次，一面參照圖22~圖27一面對高效地代入多元多項式之係數之方法進行說明。其次，一面參照圖28一面對可實現本技術之實施形態之各演算法的資訊處理裝置之硬體構成例進行說明。最後，概括本實施形態之技術思想，並對根據該技術思想獲得之作用效果簡單地進行說明。

(說明目錄)

1：前言

1-1：公鑰認證方式之演算法

1-2：電子署名方式之演算法

1-3：n路徑之公鑰認證方式

2：3路徑之公鑰認證方式之演算法之構成

2-1：具體之演算法之構成例

2-2：並行化演算法之構成例

2-3：基於3次多元多項式的演算法之構成例

3：5路徑之公鑰認證方式之演算法之構成

3-1：具體之演算法之構成例

3-2：並行化演算法之構成例

3-3：基於3次多元多項式的演算法之構成例

4：向電子署名方式之變形

4-1：自3路徑之公鑰認證方式向電子署名方式之變形

4-2：自5路徑之公鑰認證方式向電子署名方式之變形

5：關於署名驗證所需之記憶體量之削減方法

5-1：關於雜湊(hash)函數之結構

5-2：對基於3路徑方式的電子署名方式之應用例

5-3：對基於5路徑方式的電子署名方式之應用例

6：關於自2進數之隨機數列抽選出3進數之隨機數列之方法

6-1：抽選出方法#1(2位元地劃分)

6-2：抽選出方法#2(未劃分)

6-3：抽選出方法#3(k位元地劃分)

6-3-1：基本構成

6-3-2：追加抽選出方法

7：關於高效地代入多元多項式之係數之方法

7-1：基本之規定

7-2：資料之結構化

7-2-1：結構化方法#1

7-2-2：結構化方法#2

7-2-3：結構化方法#3

8：硬體構成例

9：概括

<1：前言>

本實施形態係關於以對多次多元聯立方程式之求解問題之困難性為安全性之依據的公鑰認證方式及電子署名方式。但，與HFE電子署名方式等先前方法不同，本實施形態係關於利用不具有高效地解開之方法(陷門)之多次多元聯立方程式的公鑰認證方式及電子署名方式。首先，關於公鑰認證方式之演算法、電子署名方式之演算法、及n路徑之公鑰認證方式，對其概要簡單地進行說明。

[1-1：公鑰認證方式之演算法]

首先，一面參照圖1一面對公鑰認證方式之演算法之概要進行說明。圖1係用以對公鑰認證方式之演算法之概要進行說明之說明圖。

公鑰認證係用於某人(證明者)利用公鑰pk及密鑰(Secret Key)sk使另一人(驗證者)認可為本人。例如證明者A之公鑰pk_A對驗證者B公開。另一方面，證明者A之密鑰sk_A由證

明者A秘密地管理。於公鑰認證之結構中，將知曉與公鑰pk_A對應之密鑰sk_A者視為證明者A本人。

為利用公鑰認證之結構向驗證者B證明出證明者A為證明者A本人，經由對話協定向驗證者B出示證明者A知曉與公鑰pk_A對應之密鑰sk_A之證據即可。而且，於向驗證者B出示證明者A知曉密鑰sk_A之證據、且驗證者B確認完畢該證據之情形時，證明者A之正當性(為本人)得到證明。

但，為擔保安全性，對公鑰認證之結構要求以下條件。

第1個條件係「於執行對話協定時使由不具有密鑰sk之偽證者進行之偽證成立之概率無限小」。將該第1個條件成立之情況稱為「健全性」。即，換言之，所謂健全性係指「不具有密鑰sk之偽證者於執行對話協定之過程中進行之偽證不會以無法忽視之概率成立」。第2個條件係「即便執行對話協定，證明者A所持有之密鑰sk_A之資訊亦完全不會洩漏至驗證者B」。將該第2個條件成立之情況稱為「零知識性」。

為安全地進行公鑰認證，必需利用具有健全性及零知識性之對話協定。假設於使用不具有健全性及零知識性之對話協定進行認證處理之情形時，無法否定偽證之可能性及密鑰之資訊洩漏之可能性，因此，即便處理本身成功地結束，亦未證明出證明者之正當性。因此，如何保證對話協定之健全性及零知識性較為重要。

(模型)

如圖1所示，於公鑰認證方式之模型中存在證明者及驗

證者之2個實體。證明者利用鑰匙生成演算法Gen生成證明者特有之密鑰sk與公鑰pk之組。其次，證明者使用利用鑰匙生成演算法Gen生成之密鑰sk與公鑰pk之組，與驗證者執行對話協定。此時，證明者係利用證明者演算法P執行對話協定。如上所述，證明者利用證明者演算法P而於對話協定中向驗證者出示擁有密鑰sk之證據。

另一方面，驗證者利用驗證者演算法V執行對話協定，且驗證該證明者是否擁有與證明者公開之公鑰對應之密鑰。即，驗證者係驗證證明者是否擁有與公鑰對應之密鑰之實體。如上述般，公鑰認證方式之模型係包含證明者及驗證者之2個實體、以及鑰匙生成演算法Gen、證明者演算法P、驗證者演算法V之3種演算法。

再者，於以下之說明中，使用「證明者」「驗證者」之表現，但該等表現僅表示實體。因此，執行鑰匙生成演算法Gen、證明者演算法P之主體係與「證明者」之實體對應之資訊處理裝置。同樣地，執行驗證者演算法V之主體係資訊處理裝置。該等資訊處理裝置之硬體構成例如如圖28所示。即，鑰匙生成演算法Gen、證明者演算法P、驗證者演算法V係基於ROM(Read Only Memory，唯讀記憶體)904、RAM(Random Access Memory，隨機存取記憶體)906、記憶部920、可移除式記錄媒體928等中記錄之程式而由CPU(Central Processing Unit，中央處理單元)902等執行。

(鑰匙生成演算法Gen)

證明者利用鑰匙生成演算法Gen。鑰匙生成演算法Gen係生成證明者特有之密鑰sk與公鑰pk之組之演算法。利用鑰匙生成演算法Gen生成之公鑰pk公開。而且，驗證者利用所公開之公鑰pk。另一方面，利用鑰匙生成演算法Gen生成之密鑰sk由證明者秘密地管理。而且，由證明者秘密地管理之密鑰sk用以對驗證者證明出證明者擁有與公鑰pk對應之密鑰sk。於形式上，鑰匙生成演算法Gen作為以安全參數(security parameter)1^{λ}(λ為0以上之整數)為輸入、且以密鑰sk及公鑰pk為輸出之演算法，可如下式(1)般表現。

[數1]

$$(sk, pk) \leftarrow Gen(1^{\lambda}) \quad \cdots (1)$$

(證明者演算法P)

證明者利用證明者演算法P。證明者演算法P係用以對驗證者證明出證明者擁有與公鑰pk對應之密鑰sk之演算法。即，證明者演算法P係以密鑰sk及公鑰pk為輸入而執行對話協定之演算法。

(驗證者演算法V)

驗證者利用驗證者演算法V。驗證者演算法V係於對話協定中驗證證明者是否擁有與公鑰pk對應之密鑰sk之演算法。驗證者演算法V係以公鑰pk為輸入、且根據對話協定之執行結果而輸出0或1(1位元)之演算法。再者，驗證者於驗證者演算法V輸出0之情形時判斷證明者不正當，於驗證者演算法V輸出1之情形時判斷證明者正當。驗證者演算法

V於形式上可如下式(2)般表現。

[數2]

$$0/1 \leftarrow V(pk) \quad \cdots (2)$$

如上所述，為實現有意義之公鑰認證，對話協定必需滿足健全性及零知識性之2個條件。然而，為證明出證明者擁有密鑰sk，證明者必需執行與密鑰sk相關之手續，將其結果通知驗證者，之後，使驗證者執行基於該通知內容之驗證。為擔保健全性，必需執行與密鑰sk相關之手續。另一方面，必需使密鑰sk之資訊完全未洩漏至驗證者。因此，為滿足上述必要條件，必需巧妙地設計上述鑰匙生成演算法Gen、證明者演算法P、驗證者演算法V。

以上，對公鑰認證方式之演算法之概要進行了說明。

[1-2：電子署名方式之演算法]

其次，一面參照圖2一面對電子署名方式之演算法之概要進行說明。圖2係用以對電子署名方式之演算法之概要進行說明之說明圖。

與書面文件不同，無法對某經電子化之資料簽章或記載署名。因此，為證明經電子化之資料之製作者，需要可獲得與對書面文件籤章或記載署名同等之效果的電子結構。該結構為電子署名。所謂電子署名係指將僅資料之製作者知曉之署名資料與資料建立關聯並提供至收領者、且於收領者側驗證該署名資料之結構。

(模型)

如圖2所示，於電子署名方式之模型中存在署名者及驗證者之2個實體。而且，電子署名方式之模型係包含鑰匙生成演算法Gen、署名生成演算法Sig、署名驗證演算法Ver之3種演算法。

署名者利用鑰匙生成演算法Gen生成署名者特有之署名鑰匙sk與驗證鑰匙pk之組。又，署名者利用署名生成演算法Sig生成對文件M賦予之電子署名σ。即，署名者係對文件M賦予電子署名之實體。另一方面，驗證者利用署名驗證演算法Ver驗證對文件M賦予之電子署名σ。即，驗證者係為確認文件M之製作者是否為署名者而驗證電子署名σ之實體。

再者，於以下之說明中，使用「署名者」「驗證者」之表現，但該等表現僅表示實體。因此，執行鑰匙生成演算法Gen、署名生成演算法Sig之主體係與「署名者」之實體對應之資訊處理裝置。同樣地，執行署名驗證演算法Ver之主體係資訊處理裝置。該等資訊處理裝置之硬體構成例如如圖28所示。即，鑰匙生成演算法Gen、署名生成演算法Sig、署名驗證演算法Ver係基於ROM 904、RAM 906、記憶部920、可移除式記錄媒體928等中記錄之程式而由CPU 902等執行。

(鑰匙生成演算法Gen)

署名者利用鑰匙生成演算法Gen。鑰匙生成演算法Gen係生成署名者特有之署名鑰匙sk與驗證鑰匙pk之組之演算法。利用鑰匙生成演算法Gen生成之驗證鑰匙pk公開。另

一方面，利用鑰匙生成演算法Gen生成之署名鑰匙sk由署名者秘密地管理。而且，署名鑰匙sk用於對文件M賦予之電子署名σ之生成。例如鑰匙生成演算法Gen係以安全參數1^{λ}(λ為0以上之整數)為輸入、且以署名鑰匙sk及公鑰pk為輸出。於該情形時，鑰匙生成演算法Gen於形式上可如下式(3)般表現。

[數3]

$$(sk, pk) \leftarrow Gen(1^{\lambda}) \quad \cdots (3)$$

(署名生成演算法Sig)

署名者利用署名生成演算法Sig。署名生成演算法Sig係生成對文件M賦予之電子署名σ之演算法。署名生成演算法Sig係以署名鑰匙sk及文件M為輸入、且以電子署名σ為輸出之演算法。該署名生成演算法Sig於形式上可如下式(4)般表現。

[數4]

$$\sigma \leftarrow Sig(sk, M) \quad \cdots (4)$$

(署名驗證演算法Ver)

驗證者利用署名驗證演算法Ver。署名驗證演算法Ver係驗證電子署名σ是否為對文件M的正當之電子署名之演算法。署名驗證演算法Ver係以署名者之驗證鑰匙pk、文件M、電子署名σ為輸入、且以0或1(1位元)為輸出之演算法。該署名驗證演算法Ver於形式上可如下式(5)般表現。

再者，驗證者於署名驗證演算法Ver輸出0之情形時(公開鑰匙公鑰pk拒絕文書文件M及電子署名σ之情形時)判斷電子署名σ不正當，於署名驗證演算法Ver輸出1之情形時(公開鑰匙公鑰pk受理文書文件M及電子署名σ之情形時)判斷電[數署]名σ正當。

$$0/1 \leftarrow Ver(pk, M, \sigma)$$

…(5)

以上，對電子署名方式之演算法之概要進行了說明。

[1-3：n路徑之公鑰認證方式]

其次，一面參照圖3一面對n路徑之公鑰認證方式進行說明。圖3係用以對n路徑之公鑰認證方式進行說明之說明圖。

如上所述，公鑰認證方式係於對話協定中證明者向驗證者證明擁有與公鑰pk對應之密鑰sk之認證方式。又，對話協定必需滿足健全性及零知識性之2個條件。因此，於對話協定中，如圖3所示，證明者及驗證者之兩者分別一面執行處理一面進行n次資訊交換。

於n路徑之公鑰認證方式之情形時，證明者利用證明者演算法P執行處理(步驟#1)，且將資訊T_1發送至驗證者。其次，驗證者利用驗證者演算法V執行處理(步驟#2)，且將資訊T_2發送至證明者。進而，針對k=3~n，依序進行處理之執行及資訊T_k之發送，最後執行處理(步驟#n+1)。如上述般，將發送接收n次資訊之方式稱為「n路徑」之公鑰認證方式。

以上，對n路徑之公鑰認證方式進行了說明。

<2：3路徑之公鑰認證方式之演算法之構成>

以下，對3路徑之公鑰認證方式之演算法進行說明。再者，於以下之說明中，存在將3路徑之公鑰認證方式稱為「3路徑方式」之情形。

[2-1：具體之演算法之構成例(圖4)]

首先，一面參照圖4一面對3路徑方式之具體之演算法之構成例進行介紹。圖4係用以對3路徑方式之具體之演算法之構成進行說明之說明圖。此處，考慮利用2次多項式之組$(f_1(x)，\ldots，f_m(x))$作為公鑰pk之一部分之情形。其中，2次多項式$f_i(x)$設為如下式(6)般表現者。又，將向量$(x_1，\ldots，x_n)$記作x，將2次多項式之組$(f_1(x)，\ldots，f_m(x))$記作多元多項式F(x)。

[數6]

$$f_i(x_1,\cdots,x_n)=\sum_{j,k}a_{ijk}x_jx_k+\sum_j b_{ij}x_j$$

…(6)

又，2次多項式之組$(f_1(x)，\ldots，f_m(x))$可如下式(7)般表現。又，$A_1，\ldots，A_m$為n×n矩陣。進而，$b_1，\ldots，b_m$分別為n×1向量。

[數 7]

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

…（7）

若使用該表現，則多元多項式F可如下式(8)及式(9)般表現。該表現成立可根據下式(10)容易地得到確認。

[數 8]

$$F(x+y)=F(x)+F(y)+G(x,y)$$

…（8）

$$G(x,y)=\begin{pmatrix} y^T(A_1^T+A_1)x \\ \vdots \\ y^T(A_m^T+A_m)x \end{pmatrix}$$

…（9）

$$\begin{aligned} f_l(x+y) &= (x+y)^T A_l(x+y)+b_l^T(x+y) \\ &= x^T A_l x+x^T A_l y+y^T A_l x+y^T A_l y+b_l^T x+b_l^T y \\ &= f_l(x)+f_l(y)+x^T A_l y+y^T A_l x \\ &= f_l(x)+f_l(y)+x^T(A_l^T)^T y+y^T A_l x \\ &= f_l(x)+f_l(y)+(A_l^T x)^T y+y^T A_l x \end{aligned}$$

$$= f_l(x) + f_l(y) + y^T(A_l^T x) + y^T A_l x$$
$$= f_l(x) + f_l(y) + y^T(A_l^T + A_l)x$$

…(10)

如上述般，於將F(x+y)分為與x相關之第1部分、與y相關之第2部分、及與x及y之兩者相關之第3部分時，與第3部分對應之項G(x，y)成為關於x及y之雙線性。以下，存在將項G(x，y)稱為雙線性項之情形。若利用該性質，則可構築高效之演算法。

例如利用向量$t_0 \in K^n$、$e_0 \in K^m$，而將用於多元多項式F(x+r)之遮蔽的多元多項式$F_1(x)$表現為$F_1(x)=G(x，t_0)+e_0$。於該情形時，多元多項式$F(x+r_0)$與G(x)之和係如下式(11)般表現。此處，若$t_1=r_0+t_0$、$e_1=F(r_0)+e_0$，則多元多項式$F_2(x)=F(x+r_0)+F_1(x)$可藉由向量$t_1 \in K^n$、$e_1 \in K^m$表現。因此，若設定為$F_1(x)=G(x，t_0)+e_0$，則可利用K^n上之向量及K^m上之向量表現F_1及F_2，從而可實現通訊所需之資料尺寸較少之高效之演算法。

[數9]

$$F(x+r_0)+F_1(x)$$
$$= F(x)+F(r_0)+G(x,r_0)+G(x,t_0)+e_0$$
$$= F(x)+G(x,r_0+t_0)+F(r_0)+e_0$$

…(11)

再者，根據F_2(或F_1)，與r_0相關之資訊完全不會洩漏。例如即便提供e_1及t_1(或e_0及t_0)，只要未知曉e_0及t_0(或e_1及

t_1)，便完全無法知曉r_0之資訊。因此，零知識性得到擔保。以下，對基於上述邏輯構築之3路徑方式之演算法進行說明。此處說明之3路徑方式之演算法係藉由如下所述之鑰匙生成演算法Gen、證明者演算法P、驗證者演算法V構成。

(鑰匙生成演算法Gen)

鑰匙生成演算法Gen生成於環K上定義之m個多元多項式$f_1(x_1, \ldots, x_n), \ldots, f_m(x_1, \ldots, x_n)$、及向量$s=(s_1, \ldots, s_n) \in K^n$。其次，鑰匙生成演算法Gen計算$y=(y_1, \ldots, y_m) \leftarrow (f_1(s), \ldots, f_m(s))$。而且，鑰匙生成演算法Gen係將$(f_1(x_1, \ldots, x_n), \ldots, f_m(x_1, \ldots, x_n), y)$設定為公鑰pk，將s設定為密鑰。

(證明者演算法P、驗證者演算法V)

以下，一面參照圖4一面對在對話協定中證明者演算法P執行之處理及驗證者演算法V執行之處理進行說明。於該對話協定中，證明者完全不將密鑰s之資訊洩漏至驗證者，而向驗證者表明「自身知曉滿足$y=F(s)$之s」。另一方面，驗證者驗證證明者是否知曉滿足$y=F(s)$之s。再者，公鑰pk對驗證者公開。又，密鑰s由證明者秘密地管理。以下，按照圖4所示之流程圖進行說明。

步驟#1：

如圖4所示，首先，證明者演算法P隨機地生成向量r_0、$t_0 \in K^n$及$e_0 \in K^m$。其次，證明者演算法P計算$r_1 \leftarrow s-r_0$。該計算相當於利用向量r_0遮蔽密鑰s之操作。進而，證明者演算

法P計算$t_1 \leftarrow r_0 - t_0$。其次，證明者演算法P計算$e_1 \leftarrow F(r_0) - e_0$。

步驟#1(續)：

其次，證明者演算法P計算$c_0 \leftarrow H(r_1, G(t_0, r_1) + e_0)$。其次，證明者演算法P計算$c_1 \leftarrow H(t_0, e_0)$。其次，證明者演算法P計算$c_2 \leftarrow H(t_1, e_1)$。將於步驟#1中生成之訊息$(c_0, c_1, c_2)$發送至驗證者演算法V。

步驟#2：

收到訊息(c_0, c_1, c_2)之驗證者演算法V自3個驗證圖案中選擇利用哪一驗證圖案。例如驗證者演算法V自表示驗證圖案之種類之3個數值{0，1，2}中選擇1個數值，且將所選擇之數值設定為要求Ch。將該要求Ch發送至證明者演算法P。

步驟#3：

收到要求Ch之證明者演算法P根據所收到之要求Ch生成向驗證者演算法V發送之回答Rsp。於Ch=0之情形時，證明者演算法P生成回答$Rsp=(r_0, t_1, e_1)$。於Ch=1之情形時，證明者演算法P生成回答$Rsp=(r_1, t_0, e_0)$。於Ch=2之情形時，證明者演算法P生成回答$Rsp=(r_1, t_1, e_1)$。將於步驟#3中生成之回答Rsp發送至驗證者演算法V。

步驟#4：

收到回答Rsp之驗證者演算法V利用所收到之回答Rsp而執行以下驗證處理。

於Ch=0之情形時，驗證者演算法V驗證$c_1=H(r_0-t_1,$

$F(r_0)-e_1$)之等號是否成立。進而，驗證者演算法V驗證$c_2=H(t_1, e_1)$之等號是否成立。驗證者演算法V於上述驗證全部成功之情形時輸出表示認證成功之值1，於在驗證中發生失敗之情形時輸出表示認證失敗之值0。

於Ch=1之情形時，驗證者演算法V驗證$c_0=H(r_1, G(t_0, r_1)+e_0)$之等號是否成立。進而，驗證者演算法V驗證$c_1=H(t_0, e_0)$之等號是否成立。驗證者演算法V於上述驗證全部成功之情形時輸出表示認證成功之值，於在驗證中發生失敗之情形時輸出表示認證失敗之值0。

於Ch=2之情形時，驗證者演算法V驗證$c_0=H(r_1, y-F(r_1)-G(t_1, r_1)-e_1)$之等號是否成立。進而，驗證者演算法V驗證$c_2=H(t_1, e_1)$之等號是否成立。驗證者演算法V於上述驗證全部成功之情形時輸出表示認證成功之值，於在驗證中存在失敗之情形時輸出表示認證失敗之值0。

以上，對3路徑方式之高效之演算法之構成例進行了說明。

[2-2：並行化演算法之構成例(圖5)]

其次，一面參照圖5一面對將圖4所示之3路徑方式之演算法並行化之方法進行說明。再者，省略鑰匙生成演算法Gen之構成之說明。

且說，若應用上述對話協定，則可將偽證成功之概率抑制為2/3以下。因此，若執行2次該對話協定，則可將偽證成功之概率抑制為$(2/3)^2$以下。進而，若執行N次該對話協定，則偽證成功之概率成為$(2/3)^N$，若使N為充分大之數

(例如N=140)，則偽證成功之概率減小至可忽視之程度。

作為執行複數次對話協定之方法，例如可考慮逐次地重複進行複數次訊息、要求、回答之交換的串列方法、及以1次交換進行複數次訊息、要求、回答之交換的並行方法。進而，亦可考慮組合串列方法與並行方法的複合型之方法。此處，一面參照圖5一面對並行執行3路徑方式之上述對話協定之演算法(以下稱為並行化演算法)進行說明。

步驟#1：

如圖5所示，首先，證明者演算法P針對i=1~N執行以下處理(1)~處理(6)。

處理(1)：證明者演算法P隨機地生成向量r_{0i}、$t_{0i}\in K^n$及$e_{0i}\in K^m$。

處理(2)：證明者演算法P計算$r_{1i}\leftarrow s-r_{0i}$。該計算相當於利用向量$r_{0i}$遮蔽密鑰s之操作。進而，證明者演算法P計算$t_{1i}\leftarrow r_{0i}+t_{0i}$。

處理(3)：證明者演算法P計算$e_{1i}\leftarrow F(r_{0i})-e_{0i}$。

處理(4)：證明者演算法P計算$c_{0i}\leftarrow H(r_{1i}, G(r_{1i}, t_{0i})+e_{0i})$。

處理(5)：證明者演算法P計算$c_{1i}\leftarrow H(t_{0i}, e_{0i})$。

處理(6)：證明者演算法P計算$c_{2i}\leftarrow H(t_{1i}, e_{1i})$。

步驟#1(續)

於針對i=1~N執行上述處理(1)~處理(6)後，證明者演算法P計算$Cmt\leftarrow H(c_{01}, c_{11}, c_{21}, \ldots, c_{0N}, c_{1N}, c_{2N})$。將於步驟#1中生成之雜湊值Cmt發送至驗證者演算法V。如上

述般，藉由將訊息(c_{01}，c_{11}，c_{21}，...，c_{0N}，c_{1N}，c_{2N})轉換為雜湊值後發送至驗證者演算法V，可削減通訊量。

步驟#2：

收到雜湊值Cmt之驗證者演算法V針對i=1~N之各者，自3個驗證圖案中選擇利用哪一驗證圖案。例如驗證者演算法V針對i=1~N之各者，自表示驗證圖案之種類之3個數值{0，1，2}中選擇1個數值，且將所選擇之數值設定為要求Ch_i。將要求Ch_1，...，Ch_N發送至證明者演算法P。

步驟#3：

收到要求Ch_1，...，Ch_N之證明者演算法P根據所收到之要求Ch_1，...，Ch_N之各者，生成向驗證者演算法V發送之回答Rsp_1，...，Rsp_N。於Ch_i=0之情形時，證明者演算法P生成$Rsp_i=(r_{0i}, t_{1i}, e_{1i}, c_{0i})$。於$Ch_i$=1之情形時，證明者演算法P生成$Rsp_i=(r_{1i}, t_{0i}, e_{0i}, c_{2i})$。於$Ch_i$=2之情形時，證明者演算法P生成$Rsp_i=(r_{1i}, t_{1i}, e_{1i}, c_{1i})$。

將於步驟#3中生成之回答Rsp_1，...，Rsp_N發送至驗證者演算法V。

步驟#4：

收到回答Rsp_1，...，Rsp_N之驗證者演算法V利用所收到之回答Rsp_1，...，Rsp_N而針對i=1~N執行以下處理(1)~處理(3)。其中，驗證者演算法V於Ch_i=0之情形時執行處理(1)，於Ch_i=1之情形時執行處理(2)，於Ch_i=2之情形時執行處理(3)。

處理(1)：於Ch_i=0之情形時，驗證者演算法V根據Rsp_i取

出 $(r_{0i}, t_{1i}, e_{1i}, c_{0i})$。其次，驗證者演算法V計算 $c_{1i}=H(r_{0i}-t_{1i}, F(r_{0i})-e_{1i})$。進而，驗證者演算法V計算 $c_{2i}=H(t_{1i}, e_{1i})$。而且，驗證者演算法V保持 (c_{0i}, c_{1i}, c_{2i})。

處理(2)：於 $Ch_i=1$ 之情形時，驗證者演算法V根據 Rsp_i 取出 $(r_{1i}, t_{0i}, e_{0i}, c_{2i})$。其次，驗證者演算法V計算 $c_{0i}=H(r_{1i}, G(t_{0i}, r_{1i})+e_{0i})$。進而，驗證者演算法V計算 $c_{1i}=H(t_{0i}, e_{0i})$。而且，驗證者演算法V保持 (c_{0i}, c_{1i}, c_{2i})。

處理(3)：於 $Ch_i=2$ 之情形時，驗證者演算法V根據 Rsp_i 取出 $(r_{1i}, t_{1i}, e_{1i}, c_{1i})$。其次，驗證者演算法V計算 $c_{0i}=H(r_{1i}, y-F(r_{1i})-G(t_{1i}, r_{1i})-e_{1i})$。進而，驗證者演算法V計算 $c_{2i}=H(t_{1i}, e_{1i})$。而且，驗證者演算法V保持 (c_{0i}, c_{1i}, c_{2i})。

於針對 $i=1\sim N$ 執行處理(1)~處理(3)後，驗證者演算法V驗證 $Cmt=H(c_{01}, c_{11}, c_{21}, \ldots, c_{0N}, c_{1N}, c_{2N})$ 之等號是否成立。驗證者演算法V於該驗證成功之情形時輸出表示認證成功之值，於在驗證中失敗之情形時輸出表示認證失敗之值0。

以上，對3路徑方式之高效率之並行化演算法之構成例進行說明。

[2-3：基於3次多元多項式的演算法之構成例]

其次，嘗試對利用以下式(12)表現之環R上之3次多項式 f_1 的高效之演算法之構築方法進行研究。以3次多項式 f_1 之

組表現之多元多項式F=(f_1，...，f_m)滿足下式(13)之關係。其中，G_x(x，y)表示關於x成為1次之項。又，G_yx，y)表示關於y成為1次之項。若表現為G_x=(gx_1，...，g_{xm})、G_y=(g_{y1}，...，g_{ym})，則g_{x1}及g_{y1}分別可如下式(14)及式(15)般展開。其中，由於g_{x1}之右邊第2項關於x及y中之任一者均為1次，故而亦可包含於g_{y1}中。

[數10]

$$f_l(x_1,\cdots,x_n)=\sum_{i,j,k}a_{lijk}x_ix_jx_k+\sum_{i,j}b_{lij}x_ix_j+\sum_i c_{li}x_i$$

…(12)

$$F(x+y)-F(x)-F(y)=G_x(x,y)+G_y(x,y)$$

…(13)

$$g_{xl}(x_1,\cdots,x_n,y_1,\cdots,y_n)=\sum_{i,j,k}\left(a_{lijk}+a_{likj}+a_{lkji}\right)y_iy_jx_k+\sum_{i,j}\left(b_{lij}+b_{lji}\right)x_iy_j$$

…(14)

$$g_{yl}(x_1,\cdots,x_n,y_1,\cdots,y_n)=\sum_{i,j,k}\left(a_{lijk}+a_{likj}+a_{lkji}\right)x_ix_jy_k$$

…(15)

如根據上式(14)及式(15)所知般，G_x(x，y)及G_y(x，y)關於x及y為加法同態(additive homomorphism)型。因此，利用該等性質，與利用2次多項式f_i的高效之演算法之構築方法同樣地，導入新變數r_0、r_1、t_0、u_0、e_0而分割公鑰

F(s)。

由於多項式G_x、G_y為加法同態型，故而利用變數r_0、r_1、t_0、u_0、e_0，而使下式(16)~式(19)之關係成立。下式(16)~式(19)可分為可以(r_0，t_0，u_0，e_0)再現之第1部分、可以(r_0，u_1，e_1)再現之第2部分、可以(r_1，t_0，e_0)再現之第3部分、及可以(r_1，t_1，u_1，e_1)再現之第4部分。

例如下式(17)中包括之「r_0，t_0」、下式(18)中包括之「u_0」、下式(19)中包括之「$F(r_0)$，$G_y(r_0$，$u_0)$，e_0」為第1部分。又，下式(19)中包括之「$G_y(r_0$，$u_1)$，e_1」為第2部分。進而，下式(16)中包括之「e_0，$G_x(r_0$，$r_1)$」為第3部分。而且，下式(16)中包括之「e_1，$F(r_1)$，$G_x(t_1$，$r_1)$」、下式(17)中包括之「t_1」、下式(18)中包括之「u_1」為第4部分。

換言之，下式(16)包括第3及第4部分，下式(17)及下式(18)包括第1及第4部分，下式(19)包括第1及第2部分。如上述般，下式(16)~式(19)分別包括2種部分。

又，根據密鑰s之定義及下式(16)~式(19)之關係，即便利用(r_0，t_0，u_0，e_0)、(r_0，u_1，e_1)、(r_1，t_0，e_0)、(r_1，t_1，u_1，e_1)中之任1者，均可保證無法獲得密鑰s。若利用該等性質，則例如可構築利用環R上之3次多項式f_1的高效之演算法(以下稱為擴充演算法)。

[數11]

$$F(r_0+r_1)=e_0+e_1+F(r_1)+G_x(t_0,r_1)+G_x(t_1,r_1) \quad \cdots(16)$$

$$r_0 = t_0 + t_1$$

…（17）

$$r_1 = u_0 + u_1$$

…（18）

$$F(r_0) + G_y(r_0, u_1) + G_y(r_0, u_0) = e_0 + e_1$$

…（19）

以下，對具體之擴充演算法之構成例進行說明。與擴充演算法之設計相關之基本之要點在於將以下式(20)~式(22)表現之訊息發送至驗證者、及進行與第1~第4部分中之任一者相關之驗證的2點。其中，若僅利用該驗證，則無法驗證第3部分中包括之「r_1」與第4部分中包括之「r_1」是否相等。同樣地，亦無法驗證第1部分中包括之「r_0」與第2部分中包括之「r_0」是否相等、第1部分中包括之「t_0，e_0」與第3部分中包括之「t_0，e_0」是否相等。進而，亦無法驗證第2部分中包括之「u_1，e_1」與第4部分中包括之「u_1，e_1」是否相等。因此，以下，對可實現上述驗證之構成例進行介紹。

[數12]

$$c_0 = H(G_x(t_0, r_1) + e_0)$$

…（20）

$$c_1 = H(t_0, u_0)$$

…（21）

$$c_2 = H(e_1 - G_y(r_0, u_1))$$

…（22）

以下，一面參照圖6一面對3路徑方式之擴充演算法之構成例進行說明。再者，省略鑰匙生成演算法Gen之構成之說明。

步驟#1：

如圖6所示，證明者演算法P隨機地生成向量r_0、t_0、$u_0 \in K^n$及$e_0 \in K^m$。其次，證明者演算法P計算$r_1 \leftarrow s-r_0$。該計算相當於利用向量r_0遮蔽密鑰s之操作。其次，證明者演算法P計算$t_1 \leftarrow r_0-t_0$。其次，證明者演算法P計算$u_1 \leftarrow r_1-u_0$。其次，證明者演算法P計算$e_1 \leftarrow F(r_0)-e_0$。

步驟#1(續)：

其次，證明者演算法P計算$c_0 \leftarrow H(r_1, G_x(t_0, r_1)+e_0)$。其次，證明者演算法P計算$c_1 \leftarrow H(r_0-t_0, u_0)$。其次，證明者演算法P計算$c_2 \leftarrow H(r_0, e_1-G_y(r_0, u_1))$。其次，證明者演算法P計算$c_3 \leftarrow H(t_0, e_0)$。其次，證明者演算法P計算$c_4 \leftarrow H(u_1, e_1)$。將於步驟#1中生成之訊息$(c_0, c_1, c_2, c_3, c_4)$發送至驗證者演算法V。

步驟#2：

收到訊息$(c_0, c_1, c_2, c_3, c_4)$之驗證者演算法V自4個驗證圖案中選擇利用哪一驗證圖案。例如驗證者演算法V自表示驗證圖案之種類之4個數值{0，1，2，3}中選擇1個數值，將所選擇之數值設定為要求Ch。將該要求Ch發送

至證明者演算法P。

步驟#3：

收到要求Ch之證明者演算法P根據所收到之要求Ch生成向驗證者演算法V發送之回答Rsp。於Ch=0之情形時，證明者演算法P生成回答Rsp=(r_0，t_0，u_0，e_0)。於Ch=1之情形時，證明者演算法P生成回答Rsp=(r_0，u_1，e_1)。於Ch=2之情形時，證明者演算法P生成回答Rsp=(r_1，t_0，e_0)。於Ch=3之情形時，證明者演算法P生成回答Rsp=(r_1，t_1，u_1，e_1)。將於步驟#3中生成之回答Rsp發送至驗證者演算法V。

步驟#4：

收到回答Rsp之驗證者演算法V利用所收到之回答Rsp執行以下驗證處理。

於Ch=0之情形時，驗證者演算法V驗證$c_1=H(r_0-t_0，u_0)$之等號是否成立。其次，驗證者演算法V驗證$c_2=H(r_0，F(r_0)+G_y(r_0，u_0)-e_0)$之等號是否成立。其次，驗證者演算法V驗證$c_3=H(t_0，e_0)$之等號是否成立。驗證者演算法V於上述驗證全部成功之情形時輸出表示認證成功之值，於在驗證中存在失敗之情形時輸出表示認證失敗之值0。

於Ch=1之情形時，驗證者演算法V驗證$c_2=H(r_0，e_1-G_y(r_0，u_1))$之等號是否成立。其次，驗證者演算法V驗證$c_4=H(u_1，e_1)$之等號是否成立。驗證者演算法V於上述驗證全部成功之情形時輸出表示認證成功之值，於在驗證中存在失敗之情形時輸出表示認證失敗之值0。

於Ch=2之情形時，驗證者演算法V驗證$c_0=H(r_1, e_0+G_x(t_0, r_1))$之等號是否成立。其次，驗證者演算法V驗證$c_3=H(t_0, e_0)$之等號是否成立。驗證者演算法V於上述驗證全部成功之情形時輸出表示認證成功之值，於在驗證中存在失敗之情形時輸出表示認證失敗之值0。

於Ch=3之情形時，驗證者演算法V驗證$c_0=H(r_1, y-F(r_1)-e_1-G_x(t_1, r_1))$之等號是否成立。其次，驗證者演算法V驗證$c_1=H(t_1, r_1-u_1)$之等號是否成立。其次，驗證者演算法V驗證$c_4=H(u_1, e_1)$之等號是否成立。驗證者演算法V於上述驗證全部成功之情形時輸出表示認證成功之值，於在驗證中存在失敗之情形時輸出表示認證失敗之值0。

以上，對3路徑方式之擴充演算法之構成例進行了說明。藉由利用3次多項式，而實現更高之安全性。

<3：5路徑之公鑰認證方式之演算法之構成>

其次，對5路徑之公鑰認證方式之演算法進行說明。再者，於以下之說明中，存在將5路徑之公鑰認證方式稱為「5路徑方式」之情形。

於3路徑方式之情形時，每1次對話協定之偽證概率為2/3，於5路徑方式之情形時，每1次對話協定之偽證概率為$1/2+1/q$。其中，q為利用之環之位數。因此，於環之位數充分大之情形時，5路徑方式可使每1次之偽證概率降低，從而可以較少之對話協定之執行次數使偽證概率充分小。

例如於欲使偽證概率為$1/2^n$以下之情形時，於3路徑方式

中，必需將對話協定執行n/(log3-1)=1.701n次以上。另一方面，於欲使偽證概率為$1/2^n$以下之情形時，於5路徑方式中，必需將對話協定執行n/(1-log(1+1/q))次以上。因此，若q=24，則實現相同安全級別(security level)所需之通訊量係5路徑方式少於3路徑方式。

[3-1：具體之演算法之構成例(圖7)]

首先，一面參照圖7一面對5路徑方式之具體之演算法之構成例進行介紹。圖7係用以對5路徑方式之具體之演算法之構成進行說明之說明圖。此處，考慮利用2次多項式之組$(f_1(x), \ldots, f_m(x))$作為公鑰pk之一部分之情形。其中，2次多項式$f_i(x)$設為如上式(6)般表現者。又，將向量$(x_1, \ldots, x_n)$記作x，將2次多項式之組$(f_1(x), \ldots, f_m(x))$記作多元多項式F(x)。

與3路徑方式之演算法同樣地，利用2個向量$t_0 \in K^n$、$e_0 \in K^m$，而將用以遮蔽多元多項式$F(x+r_0)$的多元多項式$F_1(x)$以$F_1(x)=G(x, t_0)+e_0$之方式表現。若採用該表現，則關於多元多項式$F(x+r_0)$，可獲得以下式(23)表現之關係。

[數13]

$$\begin{aligned} & Ch_A \cdot F(x+r_0)+F_1(x) \\ & = Ch_A \cdot F(x)+Ch_A \cdot F(r_0)+Ch_A \cdot G(x,r_0)+G(x,t_0)+e_0 \\ & = Ch_A \cdot F(x)+G(x,Ch_A \cdot r_0+t_0)+Ch_A \cdot F(r_0)+e_0 \end{aligned}$$

…(23)

因此，若設為$t_1=Ch_A \cdot r_0+t_0$、$e_1=Ch_A \cdot F(r_0)+e_0$，則遮蔽後之多元多項式$F_2(x)=Ch_A \cdot F(x+r_0)+F_1(x)$亦可藉由2個向量

$t_1 \in K^n$、$e_1 \in K^m$表現。由於上述原因，若設定為$F_1(x)=G(x, t_0)+e_0$，則可利用K^n上之向量及K^m上之向量表現F_1及F_2，從而可實現通訊所需之資料尺寸較少之高效之演算法。

再者，根據F_2(或F_1)，與r_0相關之資訊完全不會洩漏。例如即便提供e_1及t_1(或e_0及t_0)，只要未知曉e_0及t_0(或e_1及t_1)，便完全無法知曉r_0之資訊。因此，零知識性得到擔保。以下，對基於上述邏輯構築之5路徑方式之演算法進行說明。此處說明之5路徑方式之演算法係藉由如下所述之鑰匙生成演算法Gen、證明者演算法P、驗證者演算法V構成。

(鑰匙生成演算法Gen)

鑰匙生成演算法Gen生成於環K上定義之多元多項式$f_1(x_1, \ldots, x_n), \ldots, f_m(x_1, \ldots, x_n)$、及向量$s=(s_1, \ldots, s_n) \in K^n$。其次，鑰匙生成演算法Gen計算$y=(y_1, \ldots, y_m) \leftarrow (f_1(s), \ldots, f_m(s))$。而且，鑰匙生成演算法Gen係將$(f_1, \ldots, f_m, y)$設定為公鑰pk，將s設定為密鑰。再者，以下，將向量$(x_1, \ldots, x_n)$記作x，將多元多項式之組$(f_1(x), \ldots, f_m(x))$記作F(x)。

(證明者演算法P、驗證者演算法V)

以下，一面參照圖7一面對在對話協定中利用證明者演算法P及驗證者演算法V執行之處理進行說明。於該對話協定中，證明者完全不將密鑰s之資訊洩漏至驗證者，而向驗證者表明「自身知曉滿足$y=F(s)$之s」。另一方面，驗證者驗證證明者是否知曉滿足$y=F(s)$之s。再者，公鑰pk對驗

證者公開。又，密鑰s由證明者秘密地管理。以下，按照圖7所示之流程圖進行說明。

步驟#1：

如圖7所示，首先，證明者演算法P隨機地生成向量$r_0 \in K^n$、$t_0 \in K^n$、$e_0 \in K^m$。其次，證明者演算法P計算$r_1 \leftarrow s-r_0$。該計算相當於利用向量r0遮蔽密鑰s之操作。其次，證明者演算法P生成向量r_0、t_0、e_0之雜湊值c_0。即，證明者演算法P計算$c_0 \leftarrow H(r_0, t_0, e_0)$。其次，證明者演算法P生成$G(t_0, r_1)+e_0$及$r_1$之雜湊值$c_1$。即，證明者演算法P計算$c_0 \leftarrow H(r_1, G(t_0, r_1)+e_0)$。將於步驟#1中生成之訊息$(c_0, c_1)$發送至驗證者演算法V。

步驟#2：

收到訊息(c_0, c_1)之驗證者演算法V自存在q種之環K之根部隨機地選擇1個數Ch_A，且將所選擇之數Ch_A發送至證明者演算法P。

步驟#3：

收到數Ch_A之證明者演算法P計算$t_1 \leftarrow Ch_A \cdot r_0 - t_0$。進而，證明者演算法P計算$e_1 \leftarrow Ch_A \cdot F(r_0) - e_0$。而且，證明者演算法P將$t_1$及$e_1$發送至驗證者演算法V。

步驟#4：

收到t_1及e_1之驗證者演算法V自2個驗證圖案中選擇利用哪一驗證圖案。例如驗證者演算法V自表示驗證圖案之種類之2個數值{0，1}中選擇1個數值，將所選擇之數值設定為要求Ch_B。將該要求Ch_B發送至證明者演算法P。

步驟#5：

收到要求Ch_B之證明者演算法P根據所收到之要求Ch_B生成送回至驗證者演算法V之回答Rsp。於$Ch_B=0$之情形時，證明者演算法P生成回答$Rsp=r_0$。於$Ch_B=1$之情形時，證明者演算法P生成回答$Rsp=r_1$。將於步驟#5中生成之回答Rsp發送至驗證者演算法V。

步驟#6：

收到回答Rsp之驗證者演算法V利用所收到之回答Rsp執行以下驗證處理。

於$Ch_B=0$之情形時，驗證者演算法V執行$r_0 \leftarrow Rsp$。而且，驗證者演算法V驗證$c_0=H(r_0, Ch_A \cdot r_0-t_1, Ch_A \cdot F(r_0)-e_1)$之等號是否成立。驗證者演算法V於該驗證成功之情形時輸出表示認證成功之值，於在驗證中存在失敗之情形時輸出表示認證失敗之值0。

於$Ch_B=1$之情形時，驗證者演算法V執行$r_1 \leftarrow Rsp$。而且，驗證者演算法V驗證$c_1=H_1(r_1, Ch_A \cdot (y-F(r_1))-G(t_1, r_1)-e_1)$之等號是否成立。驗證者演算法V於該驗證成功之情形時輸出表示認證成功之值，於在驗證中存在失敗之情形時輸出表示認證失敗之值0。

以上，對5路徑方式之高效之演算法之構成例進行說明。

[3-2：並行化演算法之構成例(圖8)]

其次，一面參照圖8一面對將圖7所示之5路徑方式之演算法並行化之方法進行說明。再者，省略鑰匙生成演算法

Gen之構成之說明。

如先前所述，若應用5路徑方式之對話協定，則可將偽證成功之概率抑制為(1/2+1/q)以下。因此，若執行2次該對話協定，則可將偽證成功之概率抑制為$(1/2+1/q)^2$以下。進而，若執行N次該對話協定，則偽證成功之概率成為$(1/2+1/q)^N$，若使N為充分大之數(例如N=80)，則偽證成功之概率小至可忽視之程度。

作為執行複數次對話協定之方法，例如可考慮逐次地重複進行複數次訊息、要求、回答之交換的串列方法、及以1次交換進行複數次訊息、要求、回答之交換的並行方法。進而，亦可考慮組合串列方法與並行方法的複合型之方法。此處，對並行地執行5路徑方式之上述對話協定之演算法(以下稱為並行化演算法)進行說明。

步驟#1：

如圖8所示，首先，證明者演算法P針對i=1~N執行處理(1)~處理(4)。

處理(1)：證明者演算法P隨機地生成向量r_{0i}、$t_{0i}\in K^n$及$e_{0i}\in K^m$。

處理(2)：證明者演算法P計算$r_{1i}\leftarrow s-r_{0i}$。該計算相當於利用向量$r_{0i}$遮蔽密鑰s之操作。

處理(3)：證明者演算法P計算$c_{0i}\leftarrow H(r_{0i}, t_{0i}, e_{0i})$。

處理(4)：證明者演算法P計算$c_{1i}\leftarrow H(r_{1i}, G(t_{0i}, r_{1i})+e_{0i})$。

於針對i=1~N執行處理(1)~處理(4)後，證明者演算法P執

行雜湊值Cmt←H(c_{01}，c_{11}，...，c_{0N}，c_{1N})。而且，將於步驟#1中生成之雜湊值Cmt發送至驗證者演算法V。

步驟#2：

收到雜湊值Cmt之驗證者演算法V針對i=1~N之各者而自存在q種之環K之根部隨機地選擇1個數Ch_{Ai}，且將所選擇之數Ch_{Ai}(i=1~N)發送至證明者演算法P。

步驟#3：

收到數Ch_{Ai}(i=1~N)之證明者演算法P針對i=1~N之各者計算t_{1i}←Ch_{Ai}•r_{0i}-t_{0i}。進而，證明者演算法P針對i=1~N之各者計算e_{1i}←Ch_{Ai}•F(r_{0i})-e_{0i}。其次，證明者演算法P計算雜湊值d←H(t_{11}，e_{11}，...，t_{1N}，e_{1N})。而且，證明者演算法P將雜湊值d發送至驗證者演算法V。

步驟#4：

收到雜湊值d之驗證者演算法V針對i=1~N之各者而自2個驗證圖案中選擇利用哪一驗證圖案。例如驗證者演算法V自表示驗證圖案之種類之2個數值｛0，1｝中選擇1個數值，且將所選擇之數值設定為要求Ch_{Bi}。將要求Ch_{Bi}(i=1~N)發送至證明者演算法P。

步驟#5：

收到要求Ch_{Bi}(i=1~N)之證明者演算法P針對i=1~N根據所收到之要求Ch_{Bi}生成送回驗證者演算法V之回答Rsp_i。於Ch_{Bi}=0之情形時，證明者演算法P生成回答Rsp_i=(r_{0i}，t_{0i}，e_{0i}，c_{1i})。於Ch_{Bi}=1之情形時，證明者演算法P生成回答Rsp_i=(r_{1i}，t_{1i}，e_{1i}，c_{0i})。將於步驟#5中生成之回答

Rsp_i(i=1~N)發送至驗證者演算法V。

步驟#6：

收到回答Rsp_i(i=1~N)之驗證者演算法V利用所收到之回答Rsp_i(i=1~N)執行以下處理(1)及處理(2)。

處理(1)：於Ch_{Bi}=0之情形時，驗證者演算法V執行(r_{0i}，t_{0i}，e_{0i}，c_{1i})←Rsp_i。而且，驗證者演算法V計算c_{0i}=H(r_{0i}，t_{0i}，e_{0i})。進而，驗證者演算法V計算t_{1i}←Ch_{Ai}•r_{0i}+t_{0i}、及e_{1i}←Ch_{Ai}•F(r_{0i})-e_{0i}。而且，驗證者演算法V保持(c_{0i}，c_{1i}，t_{1i}，e_{1i})。

處理(2)：於Ch_{Bi}=1之情形時，驗證者演算法V執行(r_{1i}，t_{1i}，e_{1i}，c_{0i})←Rsp_i。而且，驗證者演算法V計算c_{1i}=H(r_{1i}，Ch_{Ai}•(y-F(r_{1i}))-G(t_{1i}，r_{1i})-e_{1i})。而且，驗證者演算法V保持(c_{0i}，c_{1i}，t_{1i}，e_{1i})。

於針對i=1~N執行處理(1)及處理(2)後，驗證者演算法V驗證Cmt=H(c_{01}，c_{11}，...，c_{0N}，c_{1N})之等號是否成立。進而，驗證者演算法V驗證d=H(t_{11}，e_{11}，...，t_{1N}，e_{1N})之等號是否成立。而且，驗證者演算法V於上述驗證全部成功之情形時輸出表示認證成功之值，於在驗證中存在失敗之情形時輸出表示認證失敗之值0。

以上，對5路徑方式之高效之並行化演算法之構成例進行了說明。

[3-3：基於3次多元多項式的演算法之構成例]

且說，與3路徑方式之情形同樣地，嘗試對利用環R上之3次多項式f_1的高效之演算法之構築方法進行研究。於如上

式(12)般表現3次多項式f_1之情形時，根據式(14)及式(15)可知$G_x(x, y)$及$G_y(x, y)$關於x及y成為線性。

因此，利用上述性質，導入新變數r_0、r_1、t_0、u_0、e_0，而分割使公鑰F(s)成為Ch_A倍而得之項。由於多項式G_x、G_y分別關於x及y為線性，故而利用變數r_0、r_1、t_0、u_0、e_0，而使下式(24)~式(27)之關係成立。又，下式(24)~式(27)可分為與Ch_A相關之第1部分、及與Ch_A無關之第2部分。其中，第1部分可以(r_1, t_1, u_1, e_1)再現。又，第2部分可以(r_0, t_1, u_1, e_1)再現。

例如下式(32)中包括之「e_0，$G_x(t_0, r_1)$」、下式(33)中包括之「t_0」、下式(26)中包括之「u_0」、下式(27)中包括之「e_0，$G_y(r_0, u_0)$」為第1部分。另一方面，下式(24)中包括之「$Ch_A \cdot F(r_0+r_1)$，e_1，$Ch_A \cdot F(r_1)$，$G_x(t_1, r_1)$」、下式(25)中包括之「$Ch_A \cdot r_0$，t_1」、下式(34)中包括之「$Ch_A \cdot r_1$，u_1」、下式(27)中包括之「$Ch_A \cdot F(r_0)$，$G_y(r_0, u_1)$，e_1」為第2部分。

又，根據密鑰s之定義及下式(24)~式(27)之關係，即便利用(r_1, t_1, u_1, e_1)、(r_0, t_1, u_1, e_1)中之任1者，均可保證無法獲得密鑰s。若利用該等性質，則例如可構築利用環R上之3次多項式f_1的高效之演算法(以下稱為擴充演算法)。

[數14]

$$Ch_A \cdot F(r_0 + r_1) = e_0 + e_1 + Ch_A \cdot F(r_1) + G_x(t_0, r_1) + G_x(t_1, r_1) \quad \cdots (24)$$

$$Ch_A \cdot r_0 = t_0 + t_1$$

…(25)

$$Ch_A \cdot r_1 = u_0 + u_1$$

…(26)

$$Ch_A \cdot F(r_0) + G_y(r_0, u_1) + G_y(r_0, u_0) = e_0 + e_1$$

…(27)

以下，對具體之擴充演算法之構成例進行說明。與擴充演算法之設計相關之基本之要點在於將以下式(28)及式(29)表現之訊息發送至驗證者、及針對驗證者所選擇之Ch_A進行關於與Ch_A相關之部分(第1部分)之驗證的2點。其中，「為防止於驗證時將於生成訊息時利用之r_0及r_1替換為其他r_0及r_1」，以下，對追加進行與r_0及r_1相關之驗證之構成例進行介紹。

[數15]

$$c_0 = H(t_0, e_0 - G_y(r_0, u_0))$$

…(28)

$$c_1 = H(u_0, G_x(t_0, r_1) + e_0)$$

…(29)

以下，一面參照圖9一面對5路徑方式之擴充演算法之基本構成進行說明。再者，省略鑰匙生成演算法Gen之構成之說明。

步驟#1：

如圖9所示，證明者演算法P隨機地生成向量r_0、t_0、

$u_0 \in K^n$及$e_0 \in K^m$。其次，證明者演算法P計算$r_1 \leftarrow s-r_0$。該計算相當於利用向量r_0遮蔽密鑰s之操作。其次，證明者演算法P計算$c_0 \leftarrow H(r_0, t_0, e_0-G_y(r_0, u_0))$。其次，證明者演算法P計算$c_1 \leftarrow H(r_1, u_0, G_x(t_0, r_1)+e_0)$。將於步驟#1中生成之訊息$(c_0, c_1)$發送至驗證者演算法V。

步驟#2：

收到訊息(c_0, c_1)之驗證者演算法V隨機地選擇數Ch_A。將該數Ch_A發送至證明者演算法P。

步驟#3：

收到數Ch_A之證明者演算法P計算$t_1 \leftarrow Ch_A \cdot r_0-t_0$。其次，證明者演算法P計算$u_1 \leftarrow Ch_A \cdot r_1-u_0$。其次，證明者演算法P計算$e_1 \leftarrow Ch_A \cdot F(r_0)+Ch_A \cdot G_y(r_0, r_1)-e_0$。將於步驟#3中生成之$(t_1, u_1, e_1)$發送至驗證者演算法V。

步驟#4：

收到(t_1, u_1, e_1)之驗證者演算法V自2個驗證圖案中選擇利用哪一驗證圖案。例如驗證者演算法V自表示驗證圖案之種類之2個數值{0，1}中選擇1個數值，且將所選擇之數值設定為要求Ch_B。將該要求Ch_B發送至證明者演算法P。

步驟#5：

收到要求Ch_B之證明者演算法P根據所收到之要求Ch_B生成向驗證者演算法V發送之回答Rsp。於$Ch_B=0$之情形時，證明者演算法P生成回答$Rsp=r_0$。於$Ch_B=1$之情形時，證明者演算法P生成回答$Rsp=r_1$。將於步驟#5中生成之回答Rsp

發送至驗證者演算法V。

步驟#6：

收到回答Rsp之驗證者演算法V利用所收到之回答Rsp執行以下驗證處理。

於$Ch_B=0$之情形時，驗證者演算法V驗證$c_0=H(r_0, Ch_A \cdot r_0-t_1, Ch_A \cdot F(r_0)+G_y(r_0, u_1)-e_1)$之等號是否成立。驗證者演算法V於該驗證成功之情形時輸出表示認證成功之值，於在驗證中存在失敗之情形時輸出表示認證失敗之值0。

於$Ch_B=1$之情形時，驗證者演算法V驗證$c_1=H(r_1, Ch_A \cdot r_1-u_1, Ch_A \cdot (y-F(r_1))-G_x(t_1, r_1)-e_1)$之等號是否成立。驗證者演算法V於該驗證成功之情形時輸出表示認證成功之值，於在驗證中存在失敗之情形時輸出表示認證失敗之值0。

以上，對5路徑方式之擴充演算法之構成例進行了說明。藉由利用3次多項式，而實現更高之安全性。

<4：向電子署名方式之變形>

其次，對使上述公鑰認證方式向電子署名方式變形之方法進行介紹。

若使公鑰認證方式之模型中之證明者與電子署名方式中之署名者對應，則容易地理解於僅使驗證者認可證明者之方面與電子署名方式之模型近似。基於此種，而對使上述公鑰認證方式向電子署名方式變形之方法進行說明。

[4-1：自3路徑之公鑰認證方式向電子署名方式之變形

(圖10)]

首先，對自3路徑之公鑰認證方式向電子署名方式之變形進行說明。

如圖10所示，以3次對話及4個步驟#1~步驟#4表現3路徑方式之高效之演算法(例如參照圖5)。

步驟#1包括針對i=1~N生成$a_i=(r_{0i}, t_{0i}, e_{0i}, r_{1i}, t_{1i}, e_{1i}, c_{0i}, c_{1i}, c_{2i})$之處理(1)、及計算$Cmt \leftarrow H(c_{01}, c_{11}, c_{21}, \ldots, c_{0N}, c_{1N}, c_{2N})$之處理(2)。將於步驟#1中利用證明者演算法P生成之Cmt發送至驗證者演算法V。

步驟#2包括選擇$Ch_1, \ldots, Ch_N$之處理。將於步驟#2中利用驗證者演算法V選擇之$Ch_1, \ldots, Ch_N$發送至證明者演算法P。

步驟#3包括利用$Ch_1, \ldots, Ch_N$及$a_1, \ldots, a_N$生成$Rsp_1, \ldots, Rsp_N$之處理。將該處理表現為$Rsp_i \leftarrow$選擇(Ch_i, a_i)。將於步驟#3中利用證明者演算法P生成之$Rsp_1, \ldots, Rsp_N$發送至驗證者演算法V。

步驟#4包括利用$Ch_1, \ldots, Ch_N$及$Rsp_1, \ldots, Rsp_N$使$c_{01}, c_{11}, c_{21}, \ldots, c_{0N}, c_{1N}, c_{2N}$重新產生之處理(1)、及利用重新產生之$c_{01}, c_{11}, c_{21}, \ldots, c_{0N}, c_{1N}, c_{2N}$驗證$Cmt=H(c_{01}, c_{11}, c_{21}, \ldots, c_{0N}, c_{1N}, c_{2N})$之處理(2)。

以上述步驟#1~步驟#4表現之公鑰認證方式之演算法變形為如圖10所示之署名生成演算法Sig及署名驗證演算法Ver。

(署名生成演算法Sig)

首先，對署名生成演算法Sig之構成進行敘述。署名生成演算法Sig包括以下處理(1)~處理(5)。

處理(1)：署名生成演算法Sig生成$a_i=(r_{0i}, t_{0i}, e_{0i}, r_{1i}, t_{1i}, e_{1i}, c_{0i}, c_{1i}, c_{2i})$。

處理(2)：署名生成演算法Sig計算$Cmt \leftarrow H(c_{01}, c_{11}, c_{21}, \ldots, c_{0N}, c_{1N}, c_{2N})$。

處理(3)：署名生成演算法Sig計算$(Ch_1, \ldots, Ch_N) \leftarrow H(M, Cmt)$。該M係將要賦予署名之文件。

處理(4)：署名生成演算法Sig計算$Rsp_i \leftarrow$選擇(Ch_i, a_i)。

處理(5)：署名生成演算法Sig將$(Cmt, Rsp_1, \ldots, Rsp_N)$設定為署名。

(署名驗證演算法Ver)

其次，對署名驗證演算法Ver之構成進行敘述。署名驗證演算法Ver包括以下處理(1)~處理(3)。

處理(1)：署名驗證演算法Ver計算$(Ch_1, \ldots, Ch_N) \leftarrow H(M, Cmt)$。

處理(2)：署名驗證演算法Ver利用$Ch_1, \ldots, Ch_N$及$Rsp_1, \ldots, Rsp_N$生成$c_{01}, c_{11}, c_{21}, \ldots, c_{0N}, c_{1N}, c_{2N}$。

處理(3)：署名驗證演算法Ver利用重新產生之$c_{01}, c_{11}, c_{21}, \ldots, c_{0N}, c_{1N}, c_{2N}$驗證$Cmt=H(c_{01}, c_{11}, c_{21}, \ldots, c_{0N}, c_{1N}, c_{2N})$。

如以上所說明般，藉由使公鑰認證方式之模型中之證明者與電子署名方式中之署名者對應，可使公鑰認證方式之演算法向電子署名方式之演算法變形。

[4-2：自5路徑之公鑰認證方式向電子署名方式之變形(圖11)]

其次，對自5路徑之公鑰認證方式向電子署名方式之變形進行說明。

如圖11所示，以5次對話及6個步驟#1~步驟#6表現5路徑方式之高效之演算法(例如參照圖8)。

步驟#1包括針對i=1~N生成$a_i=(r_{0i}, t_{0i}, e_{0i}, r_{1i}, t_{1i}, e_{1i}, c_{0i}, c_{1i})$之處理(1)、及計算$Cmt \leftarrow H(c_{01}, c_{11}, \ldots, c_{0N}, c_{1N})$之處理(2)。將於步驟#1中利用證明者演算法P生成之Cmt發送至驗證者演算法V。

步驟#2包括選擇$Ch_{A1}, \ldots, Ch_{AN}$之處理。將於步驟#2中利用驗證者演算法V選擇之$Ch_{A1}, \ldots, Ch_{AN}$發送至證明者演算法P。

步驟#3包括針對i=1~N生成$b_i=(t_{1i}, e_{1i})$之處理、及生成$d=H(t_{11}, e_{11}, \ldots, t_{1N}, e_{1N})$之處理。將於步驟#3中利用證明者演算法P生成之d發送至驗證者演算法V。

步驟#4包括選擇$Ch_{B1}, \ldots, Ch_{BN}$之處理。將於步驟#4中利用驗證者演算法V選擇之$Ch_{B1}, \ldots, Ch_{BN}$發送至證明者演算法P。

步驟#5包括利用$Ch_{B1}, \ldots, Ch_{BN}$、$a_1, \ldots, a_N$、$b_1, \ldots, b_N$生成$Rsp_1, \ldots, Rsp_N$之處理。將該處理表現為$Rsp_i \leftarrow$選擇(Ch_{Bi}, a_i, b_i)。將於步驟#5中利用證明者演算法P生成之$Rsp_1, \ldots, Rsp_N$發送至驗證者演算法V。

步驟#6包括利用$Ch_{A1}, \ldots, Ch_{AN}$、$Ch_{B1}, \ldots, Ch_{BN}$、

Rsp_1，...，Rsp_N使c_{01}，c_{11}，...，c_{0N}，c_{1N}、t_{11}，e_{11}，...，t_{1N}，e_{1N}重新產生之處理、利用重新產生之c_{01}，c_{11}，...，c_{0N}，c_{1N}驗證$Cmt=H(c_{01}, c_{11}, ..., c_{0N}, c_{1N})$之處理、及驗證$d=H(t_{11}, e_{11}, ..., t_{1N}, e_{1N})$之處理。

以上述步驟#1~步驟#6表現之公鑰認證方式之演算法變形為如圖11所示之署名生成演算法Sig及署名驗證演算法Ver。

(署名生成演算法Sig)

首先，對署名生成演算法Sig之構成進行敘述。署名生成演算法Sig包括以下處理(1)~處理(7)。

處理(1)：署名生成演算法Sig生成$a_i=(r_{0i}, t_{0i}, e_{0i}, r_{1i}, t_{1i}, e_{1i}, c_{0i}, c_{1i})$。

處理(2)：署名生成演算法Sig計算$Cmt \leftarrow H(c_{01}, c_{11}, ..., c_{0N}, c_{1N})$。

處理(3)：署名生成演算法Sig計算$(Ch_{A1}, ..., Ch_{AN}) \leftarrow H(M, Cmt)$。該M係將要賦予署名之文件。

處理(4)：署名生成演算法Sig針對i=1~N生成$b_i=(t_{1i}, e_{1i})$。進而，署名生成演算法Sig算出$d=H(t_{11}, e_{11}, ..., t_{1N}, e_{1N})$。

處理(5)：署名生成演算法Sig計算$(Ch_{B1}, ..., Ch_{BN}) \leftarrow H(M, Cmt, Ch_{A1}, ..., Ch_{AN}, d)$。再者，亦可變形為$(Ch_{B1}, ..., Ch_{BN}) \leftarrow H(Ch_{A1}, ..., Ch_{AN}, d)$。

處理(6)：署名生成演算法Sig計算$Rsp_i \leftarrow$選擇(Ch_{Bi}, a_i, b_i)。

處理(7)：署名生成演算法Sig將(Cmt，d，Rsp_1，...，Rsp_N)設定為電子署名。

(署名驗證演算法Ver)

其次，對署名驗證演算法Ver之構成進行敘述。署名驗證演算法Ver包括以下處理(1)~處理(4)。

處理(1)：署名驗證演算法Ver計算(Ch_{A1}，...，Ch_{AN})←H(M，Cmt)。

處理(2)：署名驗證演算法Ver計算(Ch_{B1}，...，Ch_{BN})←H(M，Cmt，Ch_{A1}，...，Ch_{AN}，d)。再者，於署名驗證演算法Ver執行之處理(5)中，於變形為(Ch_{B1}，...，Ch_{BN})←H(Ch_{A1}，...，Ch_{AN}，d)之情形時，署名驗證演算法Ver計算(Ch_{B1}，...，Ch_{BN})←H(Ch_{A1}，...，Ch_{AN}，d)。

處理(3)：署名驗證演算法Ver利用Ch_{A1}，...，Ch_{AN}、Ch_{B1}，...，Ch_{BN}、Rsp_1，...，Rsp_N生成t_{11}，e_{11}，...，t_{1N}，e_{1N}、c_{01}，c_{11}，...，c_{0N}，c_{1N}。

處理(4)：署名驗證演算法Ver利用重新產生之c_{01}，c_{11}，...，c_{0N}，c_{1N}驗證Cmt=H(c_{01}，c_{11}，...，c_{0N}，c_{1N})及d=H(t_{11}，e_{11}，...，t_{1N}，e_{1N})。

如以上所說明般，藉由使公鑰認證方式之模型中之證明者與電子署名方式中之署名者對應，可使公鑰認證方式之演算法向電子署名方式之演算法變形。

<5：關於署名驗證所需之記憶體量之削減方法>

且說，於上述電子署名方式之演算法中，署名驗證演算法Ver於接收電子署名整體後執行署名驗證之處理。然

而，於上述電子署名方式之情形時，電子署名之資料尺寸相對較大。因此，於利用RFID(Radio Frequency Identification，射頻識別)等僅具有較少之記憶體容量的器件進行認證之情形時，必需注意記憶體之空餘容量或認證處理中之記憶體使用率等。又，於利用不具有充分之記憶體容量之器件之情形時，亦假定無法進行認證之情形。因此，本件發明者創作了署名驗證所需之記憶體量之削減方法。

[5-1：關於雜湊函數之結構(圖12)]

首先，本件發明者著眼於雜湊函數之結構。於較多之情形時，雜湊函數具有將輸入劃分為區塊單位、而按照區塊單位依序進行處理之結構。例如於SHA-1之情形時，雜湊函數具有如圖12所示之結構。圖12所示之雜湊函數係將填充(padding)完畢之輸入M劃分為Z個區塊m_1，...，m_Z，且一面使索引j遞增，一面逐次地使區塊m_j與初始值IV或初始值CV_j一併作用於特定之函數CF，而生成雜湊值。因此，無需於獲得中間值CV_j之時間點之前利用之區塊。因此，利用該特性，創作出了有效地削減執行演算法所需之記憶體量之結構(以下稱為記憶體削減方法)。以下，對將該結構用於上述電子署名方式之方法進行說明。

[5-2：對基於3路徑方式的電子署名方式之應用例(圖14)]

首先，對將上述記憶體削減方法應用於圖10所示之基於3路徑方式的電子署名方式之演算法之方法進行說明。

(通常之安裝方法：圖13)

通常，如圖13所示，上述電子署名方式之署名驗證演算法Ver一次性地接收構成電子署名之(Cmt，Rsp_1，...，Rsp_N)(S101)。其次，署名驗證演算法Ver執行(Ch_1，...，Ch_N)←H(M，Cmt)(S102)。其次，署名驗證演算法Ver執行(c_{01}，c_{11}，c_{21}，...，c_{0N}，c_{1N}，c_{2N})←重新產生(Ch_1，...，Ch_N；Rsp_1，...，Rsp_N)(S103)。其次，署名驗證演算法Ver驗證Cmt=H(c_{01}，c_{11}，c_{21}，...，c_{0N}，c_{1N}，c_{2N})(S104)。從而結束署名驗證之一系列之處理。

(記憶體削減方法：圖14)

於通常之安裝方法之情形時，如圖13之步驟S101般，若一次性地接收電子署名，則於步驟S103之處理結束之前皆需要用以保持(Rsp_1，...，Rsp_N)之記憶體。然而，如根據圖5之演算法構成所知般，於步驟S103中執行之(c_{0i}，c_{1i}，c_{2i})之重新產生中未利用除(Ch_i；Rsp_i)以外之資訊。又，可知若考慮圖12所示之雜湊函數之結構，則可按照區塊單位分割並執行於步驟S104中執行之雜湊函數之計算。因此，將署名驗證之處理之構成改良為圖14所示之構成。

於圖14所示之構成之情形時，署名驗證演算法Ver首先僅接收包含於電子署名中之Cmt(S111)。其次，署名驗證演算法Ver執行(Ch_1，...，Ch_N)←H(M，Cmt)(S112)。其次，署名驗證演算法Ver針對i=1~N一面使i遞增一面逐次地執行步驟S113~S115之處理。

於步驟S113中，署名驗證演算法Ver接收Rsp_i(S113)。其

次，署名驗證演算法Ver利用所接收之Rsp_i執行c_{0i}，c_{1i}，c_{2i})←重新產生(Ch_i；Rsp_i)(S114)。於執行步驟S114之處理後，無需Ch_i及Rsp_i。因此，署名驗證演算法Ver於執行步驟S114之處理後，將Ch_i及Rsp_i自記憶體中刪除。

其次，署名驗證演算法Ver執行tmp_i←H_i(tmp_{i-1}；c_{0i}，c_{1i}，c_{2i})(S115)。再者，函數H_i係輸出於雜湊函數H之內部計算至c_{0i}、c_{1i}、c_{2i}時生成之中間值的函數。實際上，由於根據所選擇之雜湊函數而函數H_i之輸入尺寸不同，故而視需要而進行附加位元等適當之輸入長之修正。若利用函數H_i，則雜湊函數H係藉由包括以下所示之處理(1)~處理(3)之演算法表現。而且，tmp_N成為雜湊函數H之最終之輸出(雜湊值)。實際上，視雜湊函數之規格，而於最終處理中進行填充之追加處理。

處理(1)：tmp_0←空文字列

處理(2)：

for i=1 to N

tmp_i←H_i(tmp_{i-1}；c_{0i}，c_{1i}，c_{2i})

end for

處理(3)：output tmp_N

於針對i=1~N執行步驟S113~S115之處理後，署名驗證演算法Ver驗證Cmt=tmp_N是否成立(S116)，而結束署名驗證之一系列之處理。如上所述，署名驗證演算法Ver於重複執行步驟S113~S115之處理之過程中將不再需要之資訊自記憶體中刪除。因此，可將署名驗證所需之記憶體量抑制

為最小限度。其結果，即便為僅具有較少之記憶體容量之器件，亦可進行上述署名驗證。

[5-3：對基於5路徑方式的電子署名方式之應用例(圖16)]

其次，對將上述記憶體削減方法應用於圖11所示之基於5路徑方式的電子署名方式之演算法之方法進行說明。

(通常之安裝方法：圖15)

通常，如圖15所示，上述電子署名方式之署名驗證演算法Ver一次性地接收構成電子署名之(Cmt，d，Rsp_1，...，Rsp_N)(S121)。其次，署名驗證演算法Ver執行(Ch_{A1}，...，Ch_{AN})←H(M，Cmt)(S122)。其次，署名驗證演算法Ver執行(Ch_{B1}，...，Ch_{BN})←H(M，Cmt，Ch_{A1}，...，Ch_{AN}，d)(S123)。其次，署名驗證演算法Ver執行(c_{01}，c_{11}，...，c_{0N}，c_{1N}，d_{11}，e_{11}，...，d_{1N}，e_{1N})←重新產生(Ch_{A1}，...，Ch_{AN}，Ch_{B1}，...，Ch_{BN}；Rsp_1，...，Rsp_N)(S124)。其次，署名驗證演算法Ver驗證Cmt=H(c_{01}，c_{11}，...，c_{0N}，c_{1N})及d=H(d_{11}，e_{11}，...，d_{1N}，e_{1N})(S125)，而結束署名驗證之一系列之處理。

(記憶體削減方法：圖16)

如圖15之步驟S121般，若一次性地接收電子署名，則於步驟S124之處理結束之前皆需要用以保持(Rsp_1，...，Rsp_N)之記憶體。然而，如根據圖8之演算法構成所知般，於步驟S124中執行之(c_{0i}，c_{1i}，d_{1i}，e_{1i})之重新產生中未利用除(Ch_{Ai}，Ch_{Bi}；Rsp_i)以外之資訊。又，可知若考慮圖12

所示之雜湊函數之結構，則可按照區塊單位分割並執行步驟S125中執行之雜湊函數之計算。因此，將署名驗證之處理之構成改良為圖16所示之構成。

於圖16所示之構成之情形時，署名驗證演算法Ver首先僅接收包含於電子署名中之Cmt(S131)。其次，署名驗證演算法Ver執行$(Ch_{A1}, \ldots, Ch_{AN}) \leftarrow H(M, Cmt)$(S132)。

其次，署名驗證演算法Ver接收d(S133)。其次，署名驗證演算法Ver利用所收到之d，執行$(Ch_{B1}, \ldots, Ch_{BN}) \leftarrow H(M, Cmt, Ch_{A1}, \ldots, Ch_{AN}, d)$(S134)。於執行步驟S134之處理後，便不再需要d。因此，署名驗證演算法Ver於執行步驟S134之處理後，將d自記憶體中刪除。其次，署名驗證演算法Ver針對i=1~N，一面使i遞增一面逐次地執行步驟S135~S137之處理。

於步驟S135中，署名驗證演算法Ver接收Rsp_i(S135)。其次，署名驗證演算法Ver利用所收到之Rsp_i執行$(c_{0i}, c_{1i}, t_{1i}, e_{1i}) \leftarrow$重新產生$(Ch_{Ai}, Ch_{Bi}, Rsp_i)$(S136)。於執行步驟S136之處理後，便不再需要Ch_{Ai}、Ch_{Bi}及Rsp_i。因此，署名驗證演算法Ver於執行步驟S136之處理後，將Ch_{Ai}、Ch_{Bi}及Rsp_i自記憶體中刪除。

其次，署名驗證演算法Ver執行$tmp_i \leftarrow H_i(tmp_{i-1}; c_{0i}, c_{1i})$及$tmp_i' \leftarrow H_i(tmp_{i-1}'; t_{1i}, e_{1i})$(S137)。於針對i=1~N執行步驟S135~S137之處理後，署名驗證演算法Ver驗證$Cmt=tmp_N$及$d=tmp_N'$是否成立(S138)，而結束署名驗證之一系列之處理。如上所述，署名驗證演算法Ver於重複執

行步驟S135~S137之處理之過程中將不再需要之資訊自記憶體中刪除。因此，可將署名驗證所需之記憶體量抑制為最小限度。其結果，即便為僅具有較少之記憶體容量之器件，亦可進行上述署名驗證。

以上，對署名驗證所需之記憶體量之削減方法進行了說明。

<6：關於自2進數之隨機數列抽選出3進數之隨機數列之方法>

且說，於基於3路徑方式的公鑰認證方式之演算法中，存在生成N個以上3進數之均勻隨機數之情況。然而，生成3進數之均勻隨機數的優異之隨機數生成器並非普通者。因此，必需考慮利用生成2進數之均勻隨機數的優異之隨機數生成器生成3進數之均勻隨機數之方法。因此，本件發明者創作出了自2進數之均勻隨機數高效地生成3進數之均勻隨機數之方法。以下，對該方法詳細地進行你說明。再者，於以下之說明中，將以l進制表現(l為2或3)表示之1個數計算為1符號(symbol)。

[6-1：抽選出方法#1(2位元地劃分)(圖17)]

首先，一面參照圖17一面對將M位元之2進數每2位元地劃分而抽選出3進數之方法(以下稱為抽選出方法#1)進行介紹。如圖17所示，若每2位元地劃分2進制表現之隨機數列，則可獲得M/2個2位元之隨機數。例如若將「00」與3進數之「0」、「01」、3進數之「1」、「10」、3進數之「2」建立對應關係，則可自2位元單位之2進制表現之隨機數列

獲得3進數之隨機數列。其中，2位元之值「11」除外。即，抽選出方法#1係自以2進數2符號表現之2^2種數中抽選出以3進數1符號表現之3^1種數之方法。因此，無法抽選出N個以上之3進數之概率P_1如下式(30)所示。

[數16]

$$P_1 = \sum_{M/2-N<i\leq M/2} C_i (1/4)^i (3/4)^{M/2-i}$$

…（30）

[6-2：抽選出方法#2(未劃分)(圖18)]

其次，一面參照圖18一面對未劃分地利用2進數M符號之隨機數抽選出3進數L符號之隨機數之方法(以下稱為抽選出方法#2)進行介紹。其中，L為滿足$3^L \leqq 2^M$之最大之整數。可以2進數M符號表現之數存在2^M種。另一方面，可以3進數L符號表現之數僅存在3^L種。因此，以2進數M符號表現之2^M種數中的2^M-3^L種數未用作3進制表現之隨機數。因此，無法抽選出N個以上之3進數之概率P_2如下式(31)所示。

[數17]

$$P_2 = 1 - 3^L / 2^M$$

…（31）

[6-3：抽選出方法#3(k位元地劃分)(圖19)]

上述抽選出方法#1係以最小之劃分單位劃分2進制表現之隨機數列之方法。另一方面，上述抽選出方法#2係以

(由於可認為M位元地劃分)最大之劃分單位劃分2進制表現之隨機數列之方法。如根據上式(30)及式(31)所知般，視劃分之長度，而無法抽選出N個以上之3進數之概率不同。順便地，如圖19所示，於以k位元單位劃分2進M符號之隨機數列之情形時，無法抽選出N個以上3進數之概率P_3如下式(32)所示。

[數18]

$$P_3 = \sum_{M/k-N/L<i\leq M/k} C_i(1-3^L/2^k)^i(3^L/2^k)^{M/k-i}$$

…(32)

若可使無法抽選出N個以上3進數之概率P_3最小化，則可最高效地抽選出3進制表現之隨機數列。例如於M=512、N=140之情形時，於k=8時，概率P_3最小。

(6-3-1：基本構成(圖20))

此處，一面參照圖20一面對自2進M符號之隨機數列抽選出3進L符號之隨機數列之處理之流程進行說明。如圖20所示，首先，生成2進M符號之隨機數列(S201)。其次，將2進M符號之隨機數列劃分為k位元單位(S202)。其次，抽選出以k位元單位劃分之位元列X_{2k}中的滿足$X_{2k}\leqq 3^L$之位元列(S203)。其次，以3進制表現輸出抽選出之位元列(S204)，而結束一系列之處理。

(6-3-2：追加抽選方法(圖21))

藉由算出以上式(15)表現之概率P_3最小之劃分之長度k，

且執行圖20所示之演算法，可自2進制表現之隨機數列高效地抽選出3進制表現之隨機數列。然而，本件發明者著眼於在圖20之步驟S204中未利用滿足$X_{2k}>3^L$之位元列之方面，創作出了更高效地抽選出3進制表現之隨機數列之方法。以下，一面參照圖21一面對該方法進行說明。

該方法係利用於圖20之步驟S204中未抽選出之位元列抽選出3進制表現之符號列之方法。如圖21所示，首先，抽選出於圖20之步驟S204中未抽選出的滿足$X_{2k}>3^L$之位元列之組(例如若將位元列之組表現為$y_1y_2...y_{N'}$，則各位元列y_i滿足$3^L \leqq y_i<2^k$)(S211)。其次，自抽選出之位元列y_i分別減去3^L，而算出新位元列之組(例如若將新位元列之組表現為$z_1z_2...z_{N'}$，則各位元列$z_i=y_i-3^L$滿足$0 \leqq z_i<2^k-3^L$)(S212)。

其次，自新位元列之組抽選出位元列X成為$X<3^{L'}$者(S213)。其中，L'為滿足$3^{L'} \leqq 2^k-3^L$之最大之整數。其次，以3進制表現輸出於步驟S213中抽選出之位元列(S214)，而結束一系列之處理。藉由應用該演算法，可以$3^{L'}/(2^k-3^L)$之概率重新抽選出L'個3進數。再者，藉由遞歸地利用該方法，可進一步抽選出較多之3進數。即，同樣地，可自於步驟S213中位元列X滿足$X \geqq 3^{L'}$之位元列中抽選出3進數。

以上，對自2進數之均勻隨機數高效地生成3進數之均勻隨機數之方法進行說明。

<7：關於高效地代入多元多項式之係數之方法>

且說，至此為止，未對在證明者(或署名者)與驗證者之

間共有多元多項式之方法具體地明示。作為共有多元多項式之方法，可考慮兩者共有於生成多元多項式之係數(隨機數)時使用之種子(seed)的方法。然而，只要兩者未共有將利用共有之種子生成之隨機數用於係數之順序，便未共有多元多項式。

[7-1：基本之規定]

因此，對在證明者(或署名者)與驗證者之間進行將利用共有之種子生成之隨機數列按照哪一順序分配給多元多項式這一基本之規定。而且，於利用多元多項式時，按照該基本之規定將隨機數列用於多元多項式。若利用此種方法，則於證明者(或署名者)與驗證者之間共有多元多項式。

[7-2：資料之結構化]

其中，構成多元多項式之係數之數量巨大。於以1位元單位表現1個係數之情形時，為表現多元多項式，最低亦需要數萬位元以上之資料。因此，將數代入至多元多項式之係數之處理之負荷非常大。因此，本件發明者創作出了以特定之單位將多元多項式之係數結構化而使對係數之代入處理高效化之方法(結構化方法#1及#2)。進而，本件發明者創作出了於對相同之多元多項式之係數執行複數次代入處理之情形時使其處理效率提昇之方法(結構化方法#3)。以下，對上述方法詳細地進行說明。

(7-2-1：結構化方法#1(圖22~圖26))

首先，對結構化方法#1進行說明。如圖22及圖23所示，

結構化方法#1係將構成多元多項式之相同種類之項之係數概括為1個資料結構之方法。於圖22之例中，將多元多項式F之係數a_{1IJ}~a_{MIJ}概括為資料結構A，將係數b_{1I}~b_{MI}概括為資料結構B。又，如圖23所示，對多元多項式G亦可應用相同之方法。於該情形時，將係數$(a_{1IJ}+a_{1JI})$~$(a_{MIJ}+a_{MJI})$概括為資料結構。

於未應用結構化方法#1之情形時，包括M個N變數多項式之多元多項式F之計算係藉由如(例1)所示之演算法進行。於(例1)之情形時，必需將1位元之和(AND)演算(&)執行$2\times N\times(N+1)\times M/2$次。進而，必需將1位元之異或(XOR)演算(^)執行$1\times N\times(N+1)\times M/2$次。再者，設為輸入x為N位元。

(例1)

```
input x ;
for L=1 to M
  for I=1 to N
    for J=I to N
      [y之第L位元]^=[a_LIJ]&[x之第I位元]&[x之第J位元];
    end for
  end for
end for
output y ;
```

另一方面，於如圖22所示般將係數結構化而將所生成之隨機數每一部分地逐次用作多元多項式F之係數之情形

時，係數之代入演算法如(例2)所示。於(例2)之情形時，僅將M位元之和運算(&)執行2×N×(N+1)/2次、且將M位元之XOR運算(^)執行N×(N+1)/2次即可。再者，$a_{(1\sim M)IJ}$分別以循環(loop)之時序生成。又，亦可隨意使用係數。例如於執行N(N-1)/2次循環時，亦可每M次僅生成1次$[a_{(1\sim M)IJ}]$，而非每次均生成$[a_{(1\sim M)IJ}]$。又，於M次循環中，亦可使$[a_{(1\sim M)IJ}]$每1位元地旋轉(rotation)一面加以利用。

(例2)

input x；

for I=1 to N

for J=I to N

[y之第1~M位元]^=$[a_{(1\sim M)IJ}]$&[x之第I位元]&[x之第J位元]；

end for

end for

output y；

又，於如圖22所示般將係數結構化之情形時，亦可將應用多元多項式F之係數而得之中間之結果保持於表中。於該情形時，上述演算法如(例3)所示。再者，於序列$a_{IJ}[0][0]\sim a_{IJ}[2^k-1][2^k-1]$中分別儲存由$a_{IJ}[x_1, \ldots, x_k][z_1, \ldots, z_k]=(a_{(k(I-1)+1)(k(J-1)+1)}\&x_1\&z_1)\hat{}\ldots\hat{}(a_{(k(I-1)+1)(k(J-1)+k)}\&x_1\&z_k)\hat{}\ldots\hat{}(a_{(k(I-1)+k)(k(J-1)+1)}\&x_k\&z_1)\hat{}\ldots\hat{}(a_{(k(I-1)+k)(k(J-1)+k)}\&x_k\&z_k)$。於(例3)之情形時，僅將M位元之XOR運算(^)執行(N/k)(N/k+1)/2次即可。其中，與(例2)之演算法相比，

所需之記憶體量成為$2^{2k}/k^2$倍。

例如於k=1時，M位元XOR運算為120*119/2=7140次，所需之記憶體量為(例2)之2^2=4倍，循環次數無變化。又，於k=2時，M位元XOR運算為60*59/2=1770次，所需之記憶體量為$2^4/4$=4倍，循環次數為1/4。於k=4時，M位元XOR運算為30*29/2=435次，所需之記憶體量為$2^8/4^2$=16倍，循環次數為1/16.4。於k=6時，M位元XOR運算為20*19/2=190次，所需之記憶體量為$2^{12}/6^2$=114倍，循環次數為1/37.6。於k=8時，M位元XOR運算為15*14/2=135次，所需之記憶體為$2^{16}/8^2$=1024倍，循環次數為1/52.9。

(例3)

input x；

for I=1 to N/k

for J=I to N/k

[y之第1~M位元]^=$a_{(1\sim M)IJ}$[x之第k(I-1)+1~k位元][x之第k(J-1)+1~k位元]；

end for

end for

output y；

再者，可以說(例3)所示之方法正好為預先計算以下式(33)定義之$F_{IJ}(...)$之值、且設為序列加以保持之方法。

[數19]

$$F(x_1,\cdots,x_n)=\sum_{I,J}F_{IJ}(x_{k(I-1)+1},\cdots,x_{k(I-1)+k},x_{k(J-1)+1},\cdots,x_{k(J-1)+k})=y \quad \cdots (33)$$

其中，$F_{IJ}(x_{k(I-1)+1},\cdots,x_{k(I-1)+k},x_{k(J-1)+1},\cdots,x_{k(J-1)+k})$是表示$F(x_1,\cdots,x_N)$中，藉由$\lfloor x_{k(I-1)+1},\cdots,x_{k(I-1)+k}\rfloor$及$\lfloor x_{k(J-1)+1},\cdots,x_{k(J-1)+k}\rfloor$規定值的部分。

以上，以將結構化方法#1對多元多項式F應用之實例為例，對具體之演算法進行了說明。藉由該構成，於執行演算法時可期待處理之高速化。

(對G之應用)

至此為止，一面參照圖22一面對應用結構化方法#1計算多元多項式F之演算法進行了說明。另一方面，由於多元多項式G之各要素亦以二次形式表現，故而如圖23所示，亦可同樣地將上述結構化方法#1直接應用於多元多項式G之計算。

例如於未應用上述結構化方法#1之情形時，計算多元多項式G之演算法如下述(例1')般表現。再者，設為輸入x及y分別為N位元。

(例1')

```
input x，y；
for L=1 to M
   for I=1 to N
      for J=1 to N
         [z之第L位元]^=[a_LIJ+a_LJI]&[x之第I位元]&[y之第J位元]；
      end for
   end for
end for
```

output z；

若對上述(例1')應用上述結構化方法#1，則計算多元多項式G之演算法如下述(例2')所示。

(例2')

input x，y；

for I=1 to N

for J=1 to N

[z之第1~M位元]^=[$a_{(1\sim M)IJ}$+$a_{(1\sim M)JI}$]&[x之第I位元]&[y之第J位元]；

end or

end for

output z；

又，於將應用多元多項式G之係數而得之中間之結果保持於表中之方法之情形時，計算多元多項式G之演算法與上述(例3)對應地如下述(例3')所示。再者，於序列$a_{IJ}[0][0]\sim a_{IJ}[2^k-1][2^k-1]$中分別儲存有$a_{IJ}[x_1, \ldots, x_k][y_1, \ldots, y_k]=(a_{(k(I-1)+1)(k(J-1)+1)}\&x_1\&y_1)\hat{}\ldots\hat{}(a_{(k(I-1)+1)(k(J-1)+k)}\&x_1\&y_k)\hat{}\ldots\hat{}(a_{(k(I-1)+k)(k(J-1)+1)}\&x_k\&y_1)\hat{}\ldots\hat{}(a_{(k(I-1)+k)(k(J-1)+k)}\&x_k\&y_k)$。

於(例3')之情形時，僅將M位元之XOR運算(^)執行$(N/k)^2$次即可。其中，與(例2')之演算法相比，所需之記憶體量成為$2^{2k}/k^2$倍。

例如於k=1時，M位元XOR運算為$120^2=14400$次，所需之記憶體量為(例2')之$2^2=4$倍，循環次數無變化。又，於

k=2時，M位元XOR運算為60^2=3600次，所需之記憶體量為$2^4/4$=4倍，循環次數為1/4。於k=4時，M位元XOR運算為30^2=900次，所需之記憶體量為$2^8/4^2$=16倍，循環次數為1/16。於k=6時，M位元XOR運算為20^2=400次，所需之記憶體量為$2^{12}/6^2$=114倍，循環次數為1/36。於k=8時，M位元XOR運算為15^2=225次，所需之記憶體為$2^{16}/8^2$=1024倍，循環次數為1/64。

(例3')

input x，y；

for I=1 to N/k

for J=I to N/k

[z之第1~M位元]^=$a_{(1\sim M)IJ}$[x之第k(I-1)+1~k位元][y之第k(J-1)+1~k位元]；

end for

end for

output z；

再者，可以說(例3')所示之方法正好為預先計算以下式(34)定義之$G_{IJ}(...)$之值、且設為序列加以保持之方法。

[數20]

$$G(x_1,\cdots,x_n)=\sum_{I,J}G_{IJ}(x_{k(I-1)+1},\cdots,x_{k(I-1)+k},y_{k(J-1)+1},\cdots,y_{k(J-1)+k})=z \quad \cdots (34)$$

其中，$G_{IJ}(x_{k(I-1)+1},\cdots,x_{k(I-1)+k},y_{k(J-1)+1},\cdots,y_{k(J-1)+k})$是表示$G(x_1,\cdots,x_N,y_1,\cdots,y_N)$中，藉由$\lfloor x_{k(I-1)+1},\cdots,x_{k(I-1)+k}\rfloor$及$\lfloor y_{k(J-1)+1},\cdots,y_{k(J-1)+k}\rfloor$規定值的部分。

以上，以計算多元多項式G之實例為例，對結構化方法#1之具體之演算法之構成進行了說明。藉由該構成，於執行演算法時可期待處理之高速化。

(對3次多元多項式F之應用)

至此為止，作為結構化方法#1之具體例，對將2次多元多項式之係數結構化之方法進行了介紹，同樣地，亦可將3次多元多項式之係數結構化。於圖24之例中，將係數a_{1IJL}~a_{MIJL}概括為資料結構A，將係數b_{1IJ}~b_{MIJ}概括為資料結構B，將係數c_{1I}~c_{MI}概括為資料結構C。藉由如上述般將係數結構化，而使多元多項式之計算高效化。

例如嘗試具體地對資料結構A之計算進行考慮。於未將係數結構化之情形時，該資料結構A之計算之演算法例如如下述(例4)所示。

(例4)

```
input x;
for Q=I to M
  for I=1 to N
    for J=I to N
      for L=J to N
        [y之第Q位元]^=[a_QIJL]&[x之第I位元]&[x之第J位元]&[x之第L位元];
      end for
    end for
  end for
```

end for

output y；

另一方面，若將係數結構化，則成為如下述(例5)所示之演算法。於(例5)之情形時，僅將M位元之和運算(&)執行2×N×(N+1)×(N+2)/6次、且將M位元之XOR運算(^)執行1×N×(N+1)×(N+2)/6次即可。再者，$a_{(1\sim M)IJL}$分別以循環之時序生成。又，亦可隨意使用係數。例如於執行循環時，亦可每M次僅生成1次[$a_{(1\sim M)IJL}$]，而非每次均生成[$a_{(1\sim M)IJL}$]。又，於M次循環中，亦可一面使[$a_{(1\sim M)IJ}$]每1位元地旋轉一面加以利用。

(例5)

input x；

for I=1 to N

for J=I to N

for L=J to N

[y之第1~M位元]^=[$a_{(1\sim M)IJL}$]&[x之第I位元]&[x之第J位元]&[x之第L位元]；

end for

end for

output y；

又，於將係數結構化之情形時，亦可將應用多元多項式F之係數而得之中間之結果保持於表中。於該情形時，上述演算法如(例6)所示。再者，於序列$a_{IJL}[0][0][0]\sim a_{IJL}[2^k-1][2^k-1][2^k-1]$中分別儲存有$a_{IJL}[x_1, \ldots, x_k][z_1, \ldots, z_k][w_1, \ldots,$

$w_k]=\Sigma_{1\leqq\alpha,\beta,\gamma\leqq k}(a_{(k(I-1)+\alpha)(k(J-1)+\beta)(k(L-1)+\gamma)}\&x_\alpha\&z_\beta\&w_\gamma)$。於(例6)之情形時，僅將M位元之XOR運算(^)執行(N/k)(N/k+1)(N/k+2)/6次即可。其中，若與(例4)之演算法相比，則所需之記憶體量成為$2^{3k}/k^3$倍。

例如於k=1時，M位元XOR運算為120*121*122/6=295240次，所需之記憶體量為(例4)之2^3=8倍，循環次數無變化。又，於k=2時，M位元XOR運算為60*61*62/6=37820次，所需之記憶體量為$2^6/8$=8倍，循環次數為1/7.8。於k=4時，M位元XOR運算為30*31*32/6=4960次，所需之記憶體量為$2^{12}/4^3$=64倍，循環次數為1/59.5。於k=6時，M位元XOR運算為20*21*22/6=1540次，所需之記憶體量為$2^{18}/6^3$=1214倍，循環次數為1/192。

(例6)

```
input x;
for I=1 to N/k
   for J=I to N/k
      for L=J to N/k
         [y之第1~M位元]^=a(1~M)IJL[x之第k(I-1)+1~k位元]
[x之第k(J-1)+1~k位元][x之第k(L-1)+1~k位元];
      end for
   end for
end for
output y;
```

再者，可以說(例6)所示之方法正好為預先計算以下式

(35)定義之$F_{IJ}(...)$之值、且設為序列加以保持之方法。

[數21]

$$F(x_1,\cdots,x_n)=y$$
$$=\sum_{I,J}F_{IJ}(x_{k(I-1)+1},\cdots,x_{k(I-1)+k},x_{k(J-1)+1},\cdots,x_{k(J-1)+k},x_{k(L-1)+1},\cdots,x_{k(L-1)+k})$$
$$\cdots\quad(35)$$

其中，$F_{IJ}(x_{k(I-1)+1},\cdots,x_{k(I-1)+k},x_{k(J-1)+1},\cdots,x_{k(J-1)+k},x_{k(L-1)+1},\cdots,x_{k(L-1)+k})$是表示$F(x_1,\cdots,x_N)$中，藉由$\lfloor x_{k(I-1)+1},\cdots,x_{k(I-1)+k}\rfloor$、$\lfloor x_{k(J-1)+1},\cdots,x_{k(J-1)+k}\rfloor$及$\lfloor x_{k(L-1)+1},\cdots,x_{k(L-1)+k}\rfloor$規定值的部分。

以上，對將結構化方法#1對3次多元多項式F應用之實例進行了說明。藉由該構成，於執行演算法時可期待處理之高速化。

(對3次多元多項式G_x、G_y之應用)

與上述(例5)或(例6)同樣地，如圖25及圖26所示，亦可對3次多元多項式G_x、G_y應用結構化方法#1。於未應用結構化方法#1之情形時，3次多元多項式G_x之資料結構A之計算例如係藉由下述(例4'X)之演算法執行。

(例4'X)

```
input x，y；
for Q=1 to M
   for I=1 to N
      for J=1 to N
         for L=1 to N
            [z之第Q位元]^=[aQIJL+aQILJ+aQLJI]&[y之第I位
```

元]&[y之第J位元]&[x之第L位元]；

end for

end for

end for

end for

output z；

若對上述(例4'X)應用上述結構化方法#1，則計算多元多項式G_x之演算法如下述(例5'X)所示。又，亦可與上述(例6)同樣地將應用多元多項式G_x之係數而得之中間之結果保持於表中。

(例5'X)

input x，y；

for I=1 to N

for J=1 to N

for L=1 to N

[z之第1~M位元]^=[$a_{(1\sim M)IJL}$+$a_{(1\sim M)ILJ}$+$a_{(1\sim M)LJI}$]&[y之第I位元]&[y之第J位元]&[x之第L位元]；

end for

end for

end for

output z；

又，於未應用結構化方法#1之情形時，3次多元多項式G_y之計算例如係藉由下述(例4'Y)之演算法執行。

(例4'Y)

input x，y；

for Q=1 to M

for I=1 to N

for J=1 to N

for L=1 to N

[z之第Q位元]^=[a_{QIJL}+a_{QILJ}+a_{QLJI}]&[x之第I位元]&[x之第J位元]&[y之第L位元]；

end for

end for

end for

end for

output z；

若對上述(例4'Y)應用上述結構化方法#1，則計算多元多項式G_y之演算法如下述(例5'Y)所示。又，亦可與上述(例6)同樣地將應用多元多項式G_y之係數而得之中間之結果保持於表中。

(例5'Y)

input x，y；

for I=1 to N

for J=1 to N

for L=1 to N

[z之第1~M位元]^=[$a_{(1\sim M)IJL}$+$a_{(1\sim M)ILJ}$+$a_{(1\sim M)LJI}$]&[x之第I位元]&[x之第J位元]&[y之第L位元]；

end for

```
    end for
  end for
  output z ;
```

以上，對將結構化方法#1對3次多元多項式G_x、G_y應用之實例進行了說明。藉由該構成，於執行演算法時可期待處理之高速化。

(7-2-2：結構化方法#2(圖27))

其次，對結構化方法#2進行說明。如圖27所示，結構化方法#2係以2次式表現多元多項式、且將2次式之列或行概括為1個資料結構之方法。於圖27之例中，沿列方向概括資料結構。

於如圖27所示般將係數結構化、且將所生成之隨機數每一部分地逐次用作多元多項式之係數之情形時，係數之代入演算法如(例4)所示。於例4之情形時，僅將N位元之和運算(&)執行(N+1)×M次、將N位元之XOR運算(^)執行N×M次、且將函數Q之演算執行M次即可。

(例4)

```
  for I=1 to N
    T^=A_I&[x之第I位元] ;
  end for
  T&=x ;
  output Q(T) ;
  Q(z) {
    z=z^(z>>1) ;
```

```
    z=z^(z>>2);
    z=z^(z>>4);
    z=z^(z>>8);
    •••
    z=z^(z>>2^Log(N));
    return z&1;
}
```

又，亦可如圖27所示般將係數結構化、且將應用多元多項式之係數而得之中間之結果保持於表中。於該情形時，係數之代入演算法如(例5)所示。再者，於序列$A_I[0]$~$A_I[2^k-1]$中分別儲存由$A_I[x_1, \ldots, x_k]=(A_{(k(I-1)+1)}\&x_1)\hat{}\ldots\hat{}(A_{(k(I-1)+k)}\&x_k)$。於例5之情形時，僅將N位元之XOR運算(^)執行(N/k)×M次、且將N位元之函數Q之演算執行M次即可。其中，與(例4)之演算法相比，所需之記憶體量成為$2^k/k$倍。

例如於k=1時，N位元XOR運算為120次，所需之記憶體量為(例4)之2倍，循環次數無變化。又，於k=4時，N位元XOR運算為30次，所需之記憶體量為$2^4/4=4$倍，循環次數為1/4。於k=8時，N位元XOR運算為15次，所需之記憶體為$2^8/8=32$倍，循環次數為1/8。於k=16時，N位元XOR運算為8次，所需之記憶體為$2^{16}/16=4096$倍，循環次數為1/15。

(例5)

```
for I=1 to N/k
```

$T^{\wedge}=A_I$[x之第k(I-1)+1~k(I-1)+k位元]；

end for

T&=x；

output Q(T)；

以上，對結構化方法#2之具體之係數之代入演算法進行了說明。藉由該構成，於執行演算法時可期待處理之高速化。

(7-2-3：結構化方法#3)

其次，對結構化方法#3進行說明。結構化方法#3係如下方法，即於對相同之多元多項式進行N次(N≧2)代入處理之情形時，並行地進行N次之量的以「生成一部分係數，而進行與該部分相關之N次處理」部分為單位之逐次之處理，而非自隨機數起生成N次多項式並進行代入處理。若應用該方法，則於無法忽視隨機數生成之成本之情形時，N次之量之處理整體之產出量(throughput)得到改善。

例如於圖5所示之演算法中，於步驟#1中，一面更新引數一面重複執行N次多元多項式F及G之計算。因此，此種計算部分係以利用相同之係數重複進行演算之方式構成。構成為於利用上述(例2)演算法計算多元多項式$F(r_{0i})(i=1\sim N)$之情形時，於將N個r_{0i}全部應用於一度生成之$[a_{IJL}]$後，執行與下一$[a_{IJL}]$相關之處理。若以如上方式構成，無需生成N次相同之係數$[a_{IJL}]$即可。

以上，對結構化方法#3之具體之係數之代入演算法進行了說明。藉由該構成，N次處理中之合計之產出量得到改

善。

<8：硬體構成例(圖28)>

上述各演算法例如可利用圖28所示之資訊處理裝置之硬體構成執行。即，該各演算法之處理係藉由利用電腦程式控制圖28所示之硬體而實現。再者，該硬體之形態任意，例如其中包括個人電腦、行動電話、PHS(Personal Handy-phone System，個人手持式電話系統)、PDA(Personal Digital Assistant，個人數位助理)等行動資訊終端、遊戲機、接觸式或非接觸式之IC(Integrated Circuit，積體電路)晶片、接觸式或非接觸式之IC卡、或者各種資訊家電。其中，上述PHS係個人手持式電話系統之簡稱。又，上述PDA係個人數位助理之簡稱。

如圖28所示，該硬體主要包括CPU 902、ROM 904、RAM 906、主機匯流排(host bus)908、及橋接器(bridge)910。進而，該硬體包括外部匯流排912、介面914、輸入部916、輸出部918、記憶部920、驅動器922、連接埠924、及通訊部926。其中，上述CPU係中央處理單元之簡稱。又，上述ROM係唯讀記憶體之簡稱。而且，上述RAM係隨機存取記憶體之簡稱。

CPU 902例如作為演算處理裝置或控制裝置而發揮功能，且基於記錄於ROM 904、RAM 906、記憶部920、或可移除式記錄媒體928中之各種程式而控制各構成要素之動作整體或其一部分。ROM 904係儲存讀入至CPU 902之程式或用於演算之資料等的機構。於RAM 906中，例如暫

時或永久地儲存讀入至CPU 902之程式、或執行該程式時適當變化之各種參數等。

該等構成要素例如經由可進行高速之資料傳輸之主機匯流排908而相互連接。另一方面，主機匯流排908例如經由橋接器910而與資料傳輸速度相對低速之外部匯流排912連接。又，作為輸入部916，例如可使用滑鼠、鍵盤、觸控面板、按鈕、開關、及操作桿等。進而，作為輸入部916，亦可使用可利用紅外線或其他電波發送控制信號的遙控器(remote controller))。

輸出部918例如為CRT(Cathode Ray Tube，陰極射線管)、LCD(Liquid Crystal Display，液晶顯示裝置)、PDP(Plasma Display Panel，電漿顯示器)、或ELD(Electro-Luminescence Display，電致發光顯示器)等顯示器裝置、揚聲器、頭戴式耳機(headphone)等音頻輸出裝置、印表機(printer)、行動電話、或傳真機等可將所取得之資訊於視覺或聽覺上通知利用者之裝置。其中，上述CRT係陰極射線管之簡稱。又，上述LCD係液晶顯示裝置之簡稱。而且，上述PDP係電漿顯示器之簡稱。進而，上述ELD係電致發光顯示器之簡稱。

記憶部920係用以儲存各種資料之裝置。作為記憶部920，例如可使用硬碟驅動器(HDD，hard disk drive)等磁記憶器件、半導體記憶器件、光記憶器件、或磁光記憶器件等。其中，上述HDD係硬碟驅動器之簡稱。

驅動器922例如係讀出記錄於磁碟、光碟、磁光碟、或

半導體記憶體等可移除式記錄媒體928中之資訊、或將資訊寫入至可移除式記錄媒體928之裝置。可移除式記錄媒體928例如為DVD(Digital Versatile Disc，數位影音光碟)媒體、藍光(Blu-ray)媒體、HD DVD(High Definition Digital Versatile Disc，高清晰度數位影音光碟)媒體、各種半導體記憶媒體等。當然，可移除式記錄媒體928例如亦可為搭載有非接觸型IC晶片之IC卡、或電子機器等。其中，上述IC係積體電路之簡稱。

連接埠924例如係如USB(Universal Serial Bus，通用串列匯流排)埠、IEEE(Institute of Electrical and Electronics Engineer，電子電機工程師協會)1394埠、SCSI(Small Computer System Interface，小電腦系統介面)、RS(Recommended Standard，推薦標準)-232C埠、或光音頻端子等般之用以連接外部連接機器930之埠。外部連接機器930例如為印表機、隨身聽(portable music player)、數位相機、數位視訊攝影機、或IC記錄器等。其中，上述USB係通用串列匯流排之簡稱。又，上述SCSI係小電腦系統介面之簡稱。

通訊部926係用以與網路932連接之通訊器件，例如為有線或無線LAN(Local Area Network，區域網路)、藍牙(Bluetooth)(註冊商標)、或WUSB(Wireless USB，無線USB)用之通訊卡、光通訊用之路由器(router)、ADSL(Asymmetric Digital Subscriber Line，非對稱數位用戶線)用之路由器、或者接觸或非接觸通訊用之器件等。又，與

通訊部926連接之網路932係藉由利用有線或無線連接之網路構成，例如為因特網、家庭內LAN、紅外線通訊、可見光通訊、廣播、或衛星通訊等。其中，上述LAN係區域網路之簡稱。又，上述WUSB係無線USB之簡稱。而且，上述ADSL係非對稱數位用戶線之簡稱。

<9：概括>

最後，對本技術之實施形態之技術內容簡單地進行概括。此處所述之技術內容例如可對PC(pesonal computer，個人電腦)、行動電話、遊戲機、資訊終端、資訊家電、汽車導航系統(car navigation system)等各種資訊處理裝置進行應用。再者，以下所述之資訊處理裝置之功能可利用1台資訊處理裝置實現，亦可利用複數台資訊處理裝置實現。又，以下所述之資訊處理裝置執行處理時使用之資料記憶機構及演算處理機構可設置於該資訊處理裝置中，亦可設置於經由網路而與該資訊處理裝置連接之機器中。

上述資訊處理裝置之功能構成可以如下方式表現。例如，下述(1)之資訊處理裝置具有執行於多次多元聯立方程式之求解困難性方面置有安全性之依據的高效之公鑰認證方式或電子署名方式之演算法的功能。

(1)

一種資訊處理裝置，其包括：數生成部，其根據在執行利用包含多次多元多項式之組$F=(f_1, \ldots, f_m)$之公鑰的公鑰認證方式或電子署名方式之演算法的實體間所共有之資訊，使用特定之函數，而生成構成上述多次多元多項式之

組F之各項之係數中所使用之數；及

分配部，其將上述數生成部生成之數分配至構成要素中包含上述多次多元多項式之組F的多次多元多項式之係數；且

上述分配部將構成要素中包含上述多次多元多項式之組F的多次多元多項式之係數中、變數之組合之種類為相同項之係數群組化，而以群組單位執行分配處理。

(2)

一種資訊處理裝置，其包括：數生成部，其根據在執行利用了包含多次多元多項式之組$F=(f_1, \ldots, f_m)$之公鑰的公鑰認證方式或電子署名方式之演算法的實體間所共有之資訊，使用特定之函數，而生成構成上述多次多元多項式之組F之各項之係數中所使用之數；及

分配部，其將上述數生成部生成之數分配至構成要素中包含上述多次多元多項式之組F的多次多元多項式之係數；且

上述分配部將構成要素中包含上述多次多元多項式之組F的多次多元多項式之以2次式表現之情形時的係數矩陣以每列或每行進行群組化，而以群組單位執行分配處理。

(3)

如上述(1)或(2)之資訊處理裝置，其中上述數生成部於上述分配部對1個群組執行分配處理時，僅就屬於該1個群組之係數而生成上述數。

(4)

如上述(3)之資訊處理裝置，其中上述數生成部於僅就屬於上述1個群組之係數而生成上述數時，生成1個數，使所生成之數以每1位元旋轉，而生成所需數量之數。

(5)

如上述(1)或(2)之資訊處理裝置，其中更包括表格保持部，其將上述係數分配至與上述各群組對應之種類之項，且以表格保持將任意之數代入至該項之變數所得之值。

(6)

一種資訊處理裝置，其包括：數生成部，其根據在執行利用包含多次多元多項式之組$F=(f_1, \ldots, f_m)$之公鑰的公鑰認證方式或電子署名方式之演算法的實體間所共有之資訊，使用特定之函數，而生成與構成上述多次多元多項式之組F的各項之係數相同數量之數；及

分配部，其將按照在上述證明者與上述認證者之間預先決定之順序而由上述數生成部生成之數分配至構成要素中包含上述多次多元多項式之組F的多次多元多項式之係數。

(7)

如上述(1)至(6)中任1項之資訊處理裝置，其中上述資訊為隨機數之種子，且

上述特定之函數為利用上述種子而生成隨機數之隨機數生成器。

(8)

如上述(1)至(7)中任1項之資訊處理裝置，其中包括：訊

息生成部，其基於在環K上定義之多次多元多項式之組$F=(f_1, \ldots, f_m)$、及向量$s \in K^n$而生成訊息；

訊息提供部，其對保持上述多次多元多項式之組F及向量$y=(y_1, \ldots, y_m)=(f_1(s), \ldots, f_m(s))$之驗證者提供上述訊息；及

回答提供部，其向上述驗證者提供與上述驗證者自k種($k \geqq 3$)驗證圖案中選擇出之驗證圖案對應之回答資訊；且

上述向量s為密鑰，

上述多次多元多項式之組F及上述向量y為公鑰，

上述回答資訊係自上述隨機數之組及上述訊息中對應於上述驗證圖案而選擇之資訊，

上述訊息係藉由利用上述公鑰及上述回答資訊執行預先準備之與該回答資訊對應之驗證圖案用的演算而獲得之資訊。

(9)

如上述(1)至(7)中任1項之資訊處理裝置，其中包括：資訊保持部，其保持於環K上定義之多次多元多項式之組$F=(f_1, \ldots, f_m)$及向量$y=(y_1, \ldots, y_m)=(f_1(s), \ldots, f_m(s))$；

訊息取得部，其取得基於上述多次多元多項式之組F及向量$s \in K^n$而生成之訊息；

圖案資訊提供部，其對提供上述訊息之證明者提供自k種($k \geqq 3$)驗證圖案中隨機選擇出之1種驗證圖案之資訊；

回答取得部，其自上述證明者取得與上述所選擇之驗證圖案對應之回答資訊；及

驗證部，其基於上述訊息、上述多次多元多項式之組F、上述向量y、及上述回答資訊，驗證上述證明者是否保持有上述向量s；且

上述向量s為密鑰，

上述多次多元多項式之組F及上述向量y為公鑰，

上述訊息係藉由利用上述公鑰及上述回答資訊執行預先準備之與該回答資訊對應之驗證圖案用的演算而獲得之資訊。

(10)

如上述(1)至(7)中任1項之資訊處理裝置，其中包括：訊息生成部，其基於在環K上定義之多次多元多項式之組$F=(f_1, \ldots, f_m)$、及向量$s \in K^n$而生成訊息；

訊息提供部，其對保持上述多次多元多項式之組F及向量$y=(y_1, \ldots, y_m)=(f_1(s), \ldots, f_m(s))$之驗證者提供上述訊息；

中間資訊生成部，其使用上述驗證者隨機選擇出之第1資訊及生成上述訊息時獲得之第2資訊，生成第3資訊；

中間資訊提供部，其向上述驗證者提供上述第3資訊；及

回答提供部，其向上述驗證者提供與上述驗證者自k種($k \geqq 2$)驗證圖案中選擇出之驗證圖案對應之回答資訊；且

上述向量s為密鑰，

上述多次多元多項式之組F及上述向量y為公鑰，

上述回答資訊係自上述訊息中對應於上述驗證圖案而選

擇之資訊，

上述訊息係藉由利用上述公鑰、上述第1資訊、上述第3資訊、上述回答資訊執行預先準備之與該回答資訊對應之驗證圖案用的演算而獲得之資訊。

(11)

如上述(1)至(7)中任1項之資訊處理裝置，其中包括：資訊保持部，其保持於環K上定義之多次多元多項式之組$F=(f_1, \ldots, f_m)$及向量$y=(y_1, \ldots, y_m)=(f_1(s), \ldots, f_m(s))$；

訊息取得部，其取得基於上述多次多元多項式之組F及向量$s \in K^n$而生成之訊息；

資訊提供部，其對提供上述訊息之證明者提供隨機選擇之第1資訊；

中間資訊取得部，其取得使用上述第1資訊及生成上述訊息時獲得之第2資訊而由上述證明者生成之第3資訊；

圖案資訊提供部，其向上述證明者提供自k種($k \geqq 3$)驗證圖案中隨機選擇出之1種驗證圖案之資訊；

回答取得部，其自上述證明者取得與上述所選擇之驗證圖案對應之回答資訊；及

驗證部，其基於上述訊息、上述第1資訊、上述第3資訊、上述多次多元多項式之組F、及上述回答資訊，驗證上述證明者是否保持有上述向量s；且

上述向量s為密鑰，

上述多次多元多項式之組F及上述向量y為公鑰，

上述訊息係藉由利用上述公鑰、上述第1資訊、上述第3

資訊、上述回答資訊執行預先準備之與該回答資訊對應之驗證圖案用的演算而獲得之資訊。

(12)

如上述(1)或(2)之資訊處理裝置，其中於重複執行複數次上述演算法之情形時，上述數生成部僅生成1次上述數，上述分配部僅執行1次上述分配處理，且

上述演算法重複利用上述分配部所分配之係數。

(13)

如上述(1)至(7)中任1項之資訊處理裝置，其中包括：署名生成部，其利用於環K上定義之多次多元多項式之組$F=(f_1, \ldots, f_m)$、及署名鑰匙$s \in K^n$，生成對文件M之電子署名；及

署名提供部，其向保持上述多次多元多項式之組F及向量$y=(f_1(s), \ldots, f_m(s))$之驗證者提供上述電子署名。

(14)

一種資訊處理方法，其包括以下步驟：根據在執行利用包含多次多元多項式之組$F=(f_1, \ldots, f_m)$之公鑰的公鑰認證方式或電子署名方式之演算法的實體間所共有之資訊，使用特定之函數，而生成構成上述多次多元多項式之組F之各項之係數中所使用之數；及

將所生成之數分配至構成要素中包含上述多次多元多項式之組F的多次多元多項式之係數；且

於上述分配步驟中，將構成要素中包含上述多次多元多項式之組F的多次多元多項式之係數中、變數之組合之種

類為相同項之係數群組化，而以群組單位執行分配處理。

(15)

一種資訊處理方法，其包括以下步驟：根據在執行利用包含多次多元多項式之組F=(f_1，...，f_m)之公鑰的公鑰認證方式或電子署名方式之演算法的實體間所共有之資訊，使用特定之函數，而生成構成上述多次多元多項式之組F之各項之係數中所使用之數；及

將所生成之數分配至構成要素中包含上述多次多元多項式之組F的多次多元多項式之係數；且

於上述分配步驟中，將構成要素中包含上述多次多元多項式之組F的多次多元多項式之以2次式表現之情形時的係數矩陣以每列或每行進行群組化，而以群組單位執行分配處理。

(16)

一種資訊處理方法，其包括以下步驟：根據在執行利用包含多次多元多項式之組F=(f_1，...，f_m)之公鑰的公鑰認證方式或電子署名方式之演算法的實體間所共有之資訊，使用特定之函數，而生成與構成上述多次多元多項式之組F的各項之係數相同數量之數；及

將按照在上述證明者與上述認證者之間預先決定之順序而由上述數生成部生成之數，分配至構成要素中包含上述多次多元多項式之組F的多次多元多項式之係數。

(17)

一種程式，其係用以使電腦實現以下功能者：數生成功

能，其根據在執行利用包含多次多元多項式之組$F=(f_1, \ldots, f_m)$之公鑰的公鑰認證方式或電子署名方式之演算法的實體間所共有之資訊，使用特定之函數，而生成構成上述多次多元多項式之組F之各項之係數中所使用之數；及

分配功能，其係將上述數生成功能所生成之數分配至構成要素中包含上述多次多元多項式之組F的多次多元多項式之係數；且

上述分配功能將構成要素中包含上述多次多元多項式之組F的多次多元多項式之係數中的、變數之組合之種類為相同項之係數群組化，而以群組單位執行分配處理。

(18)

一種程式，其係用以使電腦實現以下功能者：數生成功能，其根據在執行利用包含多次多元多項式之組$F=(f_1, \ldots, f_m)$之公鑰的公鑰認證方式或電子署名方式之演算法的實體間所共有之資訊，使用特定之函數，而生成構成上述多次多元多項式之組F之各項之係數中所使用之數；及

分配功能，其係將上述數生成功能所生成之數分配至構成要素中包含上述多次多元多項式之組F的多次多元多項式之係數；且

上述分配功能將構成要素中包含上述多次多元多項式之組F的多次多元多項式之以2次式表現之情形時的係數矩陣以每列或每行進行群組化，而以群組單位執行分配處理。

(19)

一種程式，其係用以使電腦實現以下功能者：數生成功能，其根據在執行利用包含多次多元多項式之組$F=(f_1, \ldots, f_m)$之公鑰的公鑰認證方式或電子署名方式之演算法的實體間所共有之資訊，使用特定之函數，而生成與構成上述多次多元多項式之組F的各項之係數相同數量之數；及

分配功能，其將按照在上述證明者與上述認證者之間預先決定之順序而由上述數生成部生成之數，分配至構成要素中包含上述多次多元多項式之組F的多次多元多項式之係數。

(20)

一種記錄媒體，其記錄有上述(19)之程式，且可由電腦讀取。

(備考)

上述證明者演算法P、驗證者演算法V、署名生成演算法Sig、署名驗證演算法Ver係數生成部、分配部、表保持部之一例。上述證明者演算法P係訊息生成部、訊息提供部、回答提供部、中間資訊生成部、中間資訊提供部之一例。又，上述驗證者演算法V係資訊保持部、訊息取得部、圖案資訊提供部、回答取得部、驗證部、中間資訊取得部之一例。

以上，一面參照隨附圖式一面對本技術之較佳之實施形態進行了說明，當然，本技術並不限定於該例。只要為本

領域技術人員，便明瞭於申請專利範圍所記載之範疇內，可想到各種變更例或修正例，且該等內容當然亦屬於本技術之技術性範圍。

於上述說明中，介紹了使用雜湊函數H之演算法，但亦可使用承諾(commitment)函數COM代替雜湊函數H。承諾函數COM係將文字列S及隨機數ρ取為引數之函數。作為承諾函數之例，有由Shai Halevi及Silvio Micali於國際會議CRYPTO 1996上發表之方式等。

【圖式簡單說明】

圖1係用以對公鑰認證方式之演算法之構成進行說明之說明圖。

圖2係用以對電子署名方式之演算法之構成進行說明之說明圖。

圖3係用以對n路徑之公鑰認證方式之演算法之構成進行說明之說明圖。

圖4係用以對3路徑之公鑰認證方式之高效之演算法進行說明之說明圖。

圖5係用以對3路徑之公鑰認證方式之高效之演算法之並行化進行說明之說明圖。

圖6係用以對3路徑之公鑰認證方式之高效之演算法進行說明之說明圖。

圖7係用以對5路徑之公鑰認證方式之高效之演算法之構成例進行說明之說明圖。

圖8係用以對5路徑之公鑰認證方式之高效之演算法之並

行化進行說明之說明圖。

圖9係用以對5路徑之公鑰認證方式之高效之演算法之並行化進行說明之說明圖。

圖10係用以說明對3路徑之公鑰認證方式之高效之演算法を電子署名方式之演算法進行變形之方法的說明圖。

圖11係用以說明對5路徑之公鑰認證方式之高效之演算法を電子署名方式之演算法進行變形之方法的說明圖。

圖12係用以對雜湊函數之結構例進行說明之說明圖。

圖13係用以對基於3路徑方式的電子署名方式之署名驗證方法(通常之安裝方法)進行說明之說明圖。

圖14係用以對基於3路徑方式的電子署名方式之署名驗證方法(記憶體削減方法)進行說明之說明圖。

圖15係用以對基於5路徑方式的電子署名方式之署名驗證方法(通常之安裝方法)進行說明之說明圖。

圖16係用以對基於5路徑方式的電子署名方式之署名驗證方法(記憶體削減方法)進行說明之說明圖。

圖17係用以對自2進數隨機數中抽選出3進數隨機數之方法(抽選出方法#1)進行說明之說明圖。

圖18係用以對自2進數隨機數中抽選出3進數隨機數之方法(抽選出方法#2)進行說明之說明圖。

圖19係用以對自2進數隨機數中抽選出3進數隨機數之方法(抽選出方法#3)進行說明之說明圖。

圖20係用以對自2進數隨機數抽選出3進數隨機數之方法(抽選出方法#3)進行說明之說明圖。

圖21係用以對自2進數隨機數抽選出3進數隨機數之方法(抽選出方法#3)進行說明之說明圖。

圖22係用以對用以高效地代入多元多項式之係數的資料結構化方法(結構化方法#1)進行說明之說明圖。

圖23係用以對用以高效地代入多元多項式之係數的資料結構化方法(結構化方法#1)進行說明之說明圖。

圖24係用以對用以高效地代入多元多項式之係數的資料結構化方法(結構化方法#1)進行說明之說明圖。

圖25係用以對用以高效地代入多元多項式之係數的資料結構化方法(結構化方法#1)進行說明之說明圖。

圖26係用以對用以高效地代入多元多項式之係數之資料結構化方法(結構化方法#1)進行說明之說明圖。

圖27係用以對用以高效地代入多元多項式之係數的資料結構化方法(結構化方法#2)進行說明之說明圖。

圖28係用以對可執行本技術之各實施形態之演算法的資訊處理裝置之硬體構成例進行說明之說明圖。

【主要元件符號說明】

Gen	鑰匙生成演算法
P	證明者演算法
Sig	署名生成演算法
V	驗證者演算法
Ver	署名驗證演算法

七、申請專利範圍：

1. 一種資訊處理裝置，其包括：

數生成部，其根據在執行利用包含多次多元多項式之組$F=(f_1, \ldots, f_m)$之公鑰的公鑰認證方式或電子署名方式之演算法的實體間所共有之資訊，使用特定之函數，而生成構成上述多次多元多項式之組F之各項之係數中所使用之數；及

分配部，其將上述數生成部生成之數分配至構成要素中包含上述多次多元多項式之組F的多次多元多項式之係數；且

上述分配部將構成要素中包含上述多次多元多項式之組F的多次多元多項式之係數中、變數之組合之種類為相同項之係數群組化，而以群組單位執行分配處理。

2. 一種資訊處理裝置，其包括：

數生成部，其根據在執行利用包含多次多元多項式之組$F=(f_1, \ldots, f_m)$之公鑰的公鑰認證方式或電子署名方式之演算法的實體間所共有之資訊，使用特定之函數，而生成構成上述多次多元多項式之組F之各項之係數中所使用之數；及

分配部，其將上述數生成部生成之數分配至構成要素中包含上述多次多元多項式之組F的多次多元多項式之係數；且

上述分配部將構成要素中包含上述多次多元多項式之組F的多次多元多項式之以2次式表現之情形時的係數矩

陣以每列或每行進行群組化，而以群組單位執行分配處理。

3. 如請求項1之資訊處理裝置，其中上述數生成部於上述分配部對1個群組執行分配處理時，僅就屬於該1個群組之係數而生成上述數。

4. 如請求項3之資訊處理裝置，其中上述數生成部於僅就屬於上述1個群組之係數而生成上述數時，生成1個數，使所生成之數以每1位元旋轉，而生成所需數量之數。

5. 如請求項1之資訊處理裝置，其中更包括表格保持部，該表格保持部將上述係數分配至與上述各群組對應之種類之項，且以表格保持將任意之數代入至該項之變數所得之值。

6. 一種資訊處理裝置，其包括：

數生成部，其根據在執行利用包含多次多元多項式之組$F=(f_1, \ldots, f_m)$之公鑰的公鑰認證方式或電子署名方式之演算法的實體間所共有之資訊，使用特定之函數，而生成與構成上述多次多元多項式之組F的各項之係數相同數量之數；及

分配部，其將按照在證明者與認證者之間預先決定之順序而由上述數生成部生成之數，分配至構成要素中包含上述多次多元多項式之組F的多次多元多項式之係數。

7. 如請求項1之資訊處理裝置，其中上述資訊為隨機數之種子，且

上述特定之函數為利用上述種子而生成隨機數之隨機數生成器。

8. 如請求項1之資訊處理裝置，其中包括：

訊息生成部，其基於在環K上定義之多次多元多項式之組$F=(f_1, \ldots, f_m)$、及向量$s \in K^n$而生成訊息；

訊息提供部，其對保持上述多次多元多項式之組F及向量$y=(y_1, \ldots, y_m)=(f_1(s), \ldots, f_m(s))$之驗證者提供上述訊息；及

回答提供部，其向上述驗證者提供與上述驗證者自k種($k \geqq 3$)驗證圖案中選擇出之驗證圖案對應之回答資訊；且

上述向量s為密鑰，

上述多次多元多項式之組F及上述向量y為公鑰，

上述回答資訊係自上述隨機數之組及上述訊息中對應於上述驗證圖案而選擇之資訊，

上述訊息係藉由利用上述公鑰及上述回答資訊執行預先準備之與該回答資訊對應之驗證圖案用的演算而獲得之資訊。

9. 如請求項1之資訊處理裝置，其中包括：

資訊保持部，其保持於環K上定義之多次多元多項式之組$F=(f_1, \ldots, f_m)$及向量$y=(y_1, \ldots, y_m)=(f_1(s), \ldots, f_m(s))$；

訊息取得部，其取得基於上述多次多元多項式之組F及向量$s \in K^n$而生成之訊息；

圖案資訊提供部，其對提供上述訊息之證明者提供自k種(k≧3)驗證圖案中隨機選擇出之1種驗證圖案之資訊；

回答取得部，其自上述證明者取得與上述所選擇之驗證圖案對應之回答資訊；及

驗證部，其基於上述訊息、上述多次多元多項式之組F、上述向量y、及上述回答資訊，驗證上述證明者是否保持有上述向量s；且

上述向量s為密鑰，

上述多次多元多項式之組F及上述向量y為公鑰，

上述訊息係藉由利用上述公鑰及上述回答資訊執行預先準備之與該回答資訊對應之驗證圖案用的演算而獲得之資訊。

10. 如請求項1之資訊處理裝置，其中包括：

訊息生成部，其基於在環K上定義之多次多元多項式之組$F=(f_1, \ldots, f_m)$、及向量$s \in K^n$而生成訊息；

訊息提供部，其對保持上述多次多元多項式之組F及向量$y=(y_1, \ldots, y_m)=(f_1(s), \ldots, f_m(s))$之驗證者提供上述訊息；

中間資訊生成部，其使用上述驗證者隨機選擇出之第1資訊及生成上述訊息時獲得之第2資訊，生成第3資訊；

中間資訊提供部，其向上述驗證者提供上述第3資訊；及

回答提供部，其向上述驗證者提供與上述驗證者自k種(k≧2)驗證圖案中選擇出之驗證圖案對應之回答資訊；且

上述向量s為密鑰，

上述多次多元多項式之組F及上述向量y為公鑰，

上述回答資訊係自上述訊息中對應於上述驗證圖案而選擇之資訊，

上述訊息係藉由利用上述公鑰、上述第1資訊、上述第3資訊、上述回答資訊執行預先準備之與該回答資訊對應之驗證圖案用的演算而獲得之資訊。

11. 如請求項1之資訊處理裝置，其中包括：

資訊保持部，其保持於環K上定義之多次多元多項式之組$F=(f_1, \ldots, f_m)$及向量$y=(y_1, \ldots, y_m)=(f_1(s), \ldots, f_m(s))$；

訊息取得部，其取得基於上述多次多元多項式之組F及向量$s \in K^n$而生成之訊息；

資訊提供部，其對提供上述訊息之證明者提供隨機選擇之第1資訊；

中間資訊取得部，其取得使用上述第1資訊及生成上述訊息時獲得之第2資訊而由上述證明者生成之第3資訊；

圖案資訊提供部，其向上述證明者提供自k種(k≧3)驗證圖案中隨機選擇出之1種驗證圖案之資訊；

回答取得部，其自上述證明者取得與上述所選擇之驗

證圖案對應之回答資訊；及

驗證部，其基於上述訊息、上述第1資訊、上述第3資訊、上述多次多元多項式之組F、及上述回答資訊，驗證上述證明者是否保持有上述向量s；且

上述向量s為密鑰，

上述多次多元多項式之組F及上述向量y為公鑰，

上述訊息係藉由利用上述公鑰、上述第1資訊、上述第3資訊、上述回答資訊執行預先準備之與該回答資訊對應之驗證圖案用的演算而獲得之資訊。

12. 如請求項1之資訊處理裝置，其中於重複執行複數次上述演算法之情形時，上述數生成部僅生成1次上述數，上述分配部僅執行1次上述分配處理，且

上述演算法重複利用上述分配部所分配之係數。

13. 如請求項1之資訊處理裝置，其中包括：

署名生成部，其利用於環K上定義之多次多元多項式之組$F=(f_1, \ldots, f_m)$、及署名鑰匙$s \in K^n$，生成對文件M之電子署名；及

署名提供部，其向保持上述多次多元多項式之組F及向量$y=(f_1(s), \ldots, f_m(s))$之驗證者提供上述電子署名。

14. 一種資訊處理方法，其包括以下步驟：

根據在執行利用包含多次多元多項式之組$F=(f_1, \ldots, f_m)$之公鑰的公鑰認證方式或電子署名方式之演算法的實體間所共有之資訊，使用特定之函數，而生成構成上述多次多元多項式之組F之各項之係數中所使用之數；及

將所生成之數分配至構成要素中包含上述多次多元多項式之組F的多次多元多項式之係數；且

於上述分配步驟中，將構成要素中包含上述多次多元多項式之組F的多次多元多項式之係數中、變數之組合之種類為相同項之係數群組化，而以群組單位執行分配處理。

15. 一種資訊處理方法，其包括以下步驟：

根據在執行利用包含多次多元多項式之組$F=(f_1, \ldots, f_m)$之公鑰的公鑰認證方式或電子署名方式之演算法的實體間所共有之資訊，使用特定之函數，而生成構成上述多次多元多項式之組F之各項之係數中所使用之數；及

將所生成之數分配至構成要素中包含上述多次多元多項式之組F的多次多元多項式之係數；且

於上述分配步驟中，將構成要素中包含上述多次多元多項式之組F的多次多元多項式之以2次式表現之情形時的係數矩陣以每列或每行進行群組化，而以群組單位執行分配處理。

16. 一種資訊處理方法，其包括以下步驟：

根據在執行利用包含多次多元多項式之組$F=(f_1, \ldots, f_m)$之公鑰的公鑰認證方式或電子署名方式之演算法的實體間所共有之資訊，使用特定之函數，而生成與構成上述多次多元多項式之組F的各項之係數相同數量之數；及

將按照在證明者與認證者之間預先決定之順序而由上述數生成部生成之數，分配至構成要素中包含上述多次

多元多項式之組F的多次多元多項式之係數。

17. 一種程式，其係用以使電腦實現以下功能者：

數生成功能，其根據在執行利用包含多次多元多項式之組$F=(f_1, \ldots, f_m)$之公鑰的公鑰認證方式或電子署名方式之演算法的實體間所共有之資訊，使用特定之函數，而生成構成上述多次多元多項式之組F之各項之係數中所使用之數；及

分配功能，其將上述數生成功能所生成之數分配至構成要素中包含上述多次多元多項式之組F的多次多元多項式之係數；且

上述分配功能將構成要素中包含上述多次多元多項式之組F的多次多元多項式之係數中、變數之組合之種類為相同項之係數群組化，而以群組單位執行分配處理。

18. 一種程式，其係用以使電腦實現以下功能者：

數生成功能，其根據在執行利用包含多次多元多項式之組$F=(f_1, \ldots, f_m)$之公鑰的公鑰認證方式或電子署名方式之演算法的實體間所共有之資訊，使用特定之函數，而生成構成上述多次多元多項式之組F之各項之係數中所使用之數；及

分配功能，其將上述數生成功能所生成之數分配至構成要素中包含上述多次多元多項式之組F的多次多元多項式之係數；且

上述分配功能將構成要素中包含上述多次多元多項式之組F的多次多元多項式之以2次式表現之情形時的係數

矩陣以每列或每行進行群組化，而以群組單位執行分配處理。

19. 一種程式，其係用以使電腦實現以下功能者：

數生成功能，其根據在執行利用包含多次多元多項式之組$F=(f_1, \ldots, f_m)$之公鑰的公鑰認證方式或電子署名方式之演算法的實體間所共有之資訊，使用特定之函數，而生成與構成上述多次多元多項式之組F的各項之係數相同數量之數；及

分配功能，其將按照在證明者與認證者之間預先決定之順序而由上述數生成部生成之數，分配至構成要素中包含上述多次多元多項式之組F的多次多元多項式之係數。

八、圖式：

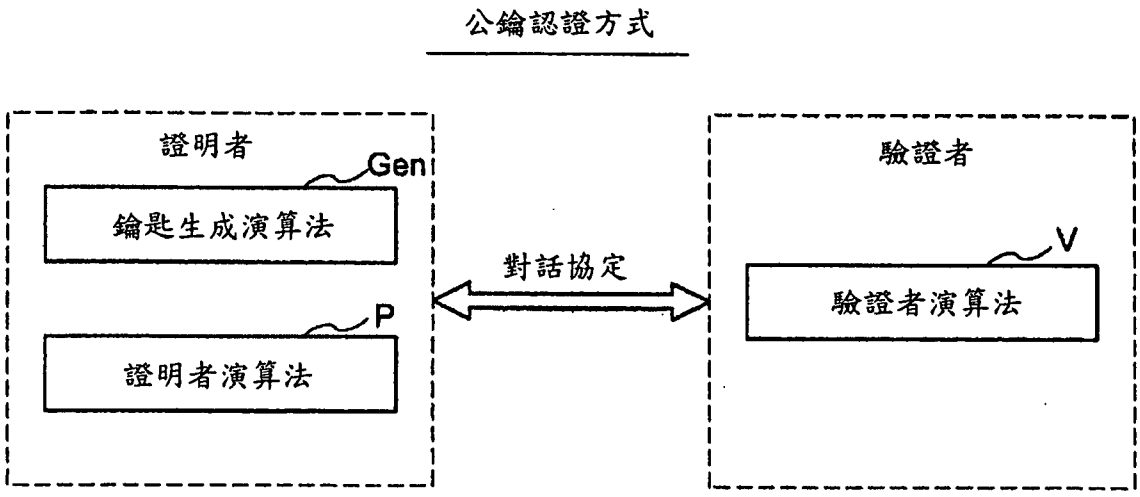

公鑰認證方式
證明者
Gen
鑰匙生成演算法
P
證明者演算法
對話協定
驗證者
V
驗證者演算法

圖1

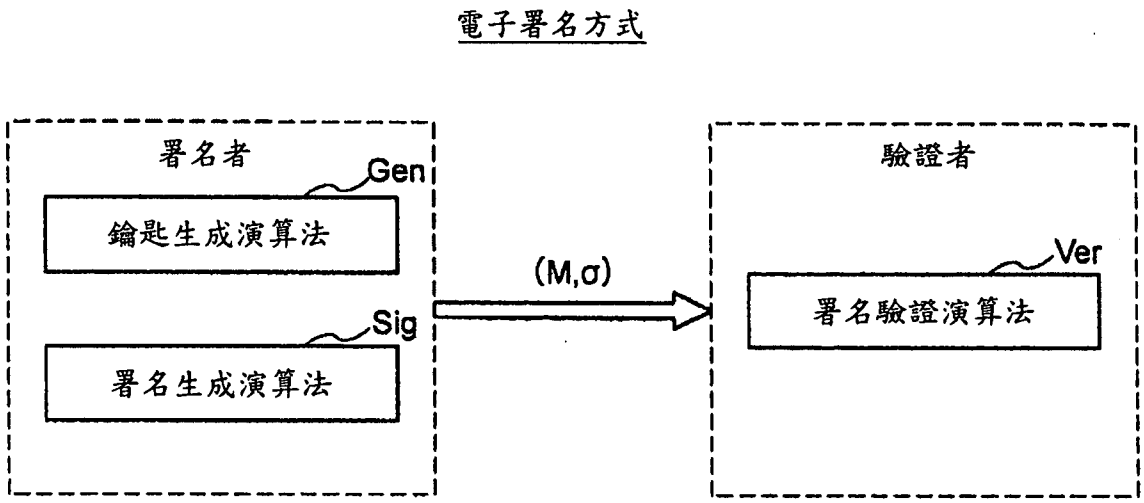

電子署名方式
署名者
Gen
鑰匙生成演算法
Sig
署名生成演算法
(M,σ)
驗證者
Ver
署名驗證演算法

圖2

n路徑之公鑰認證方式

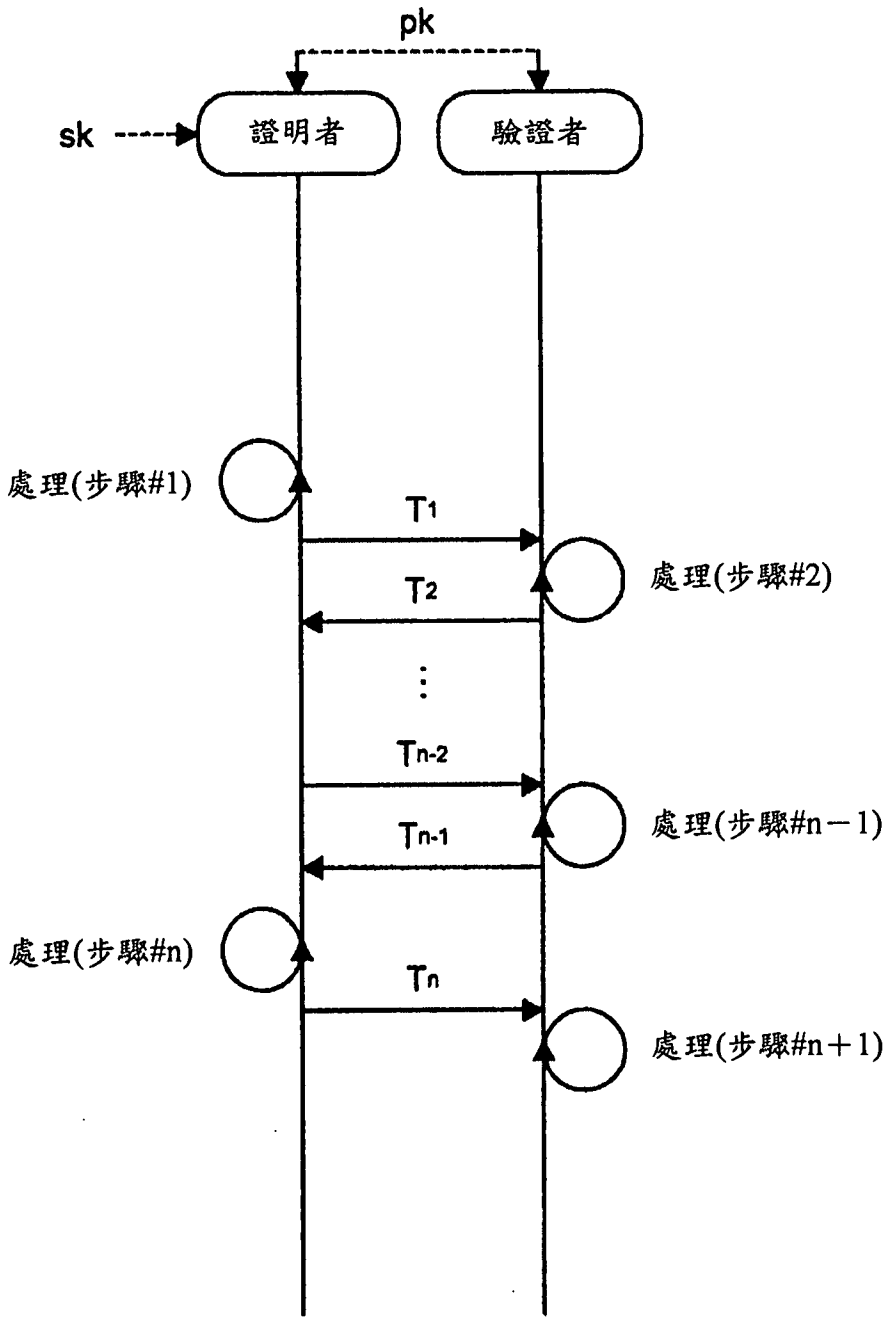


圖3

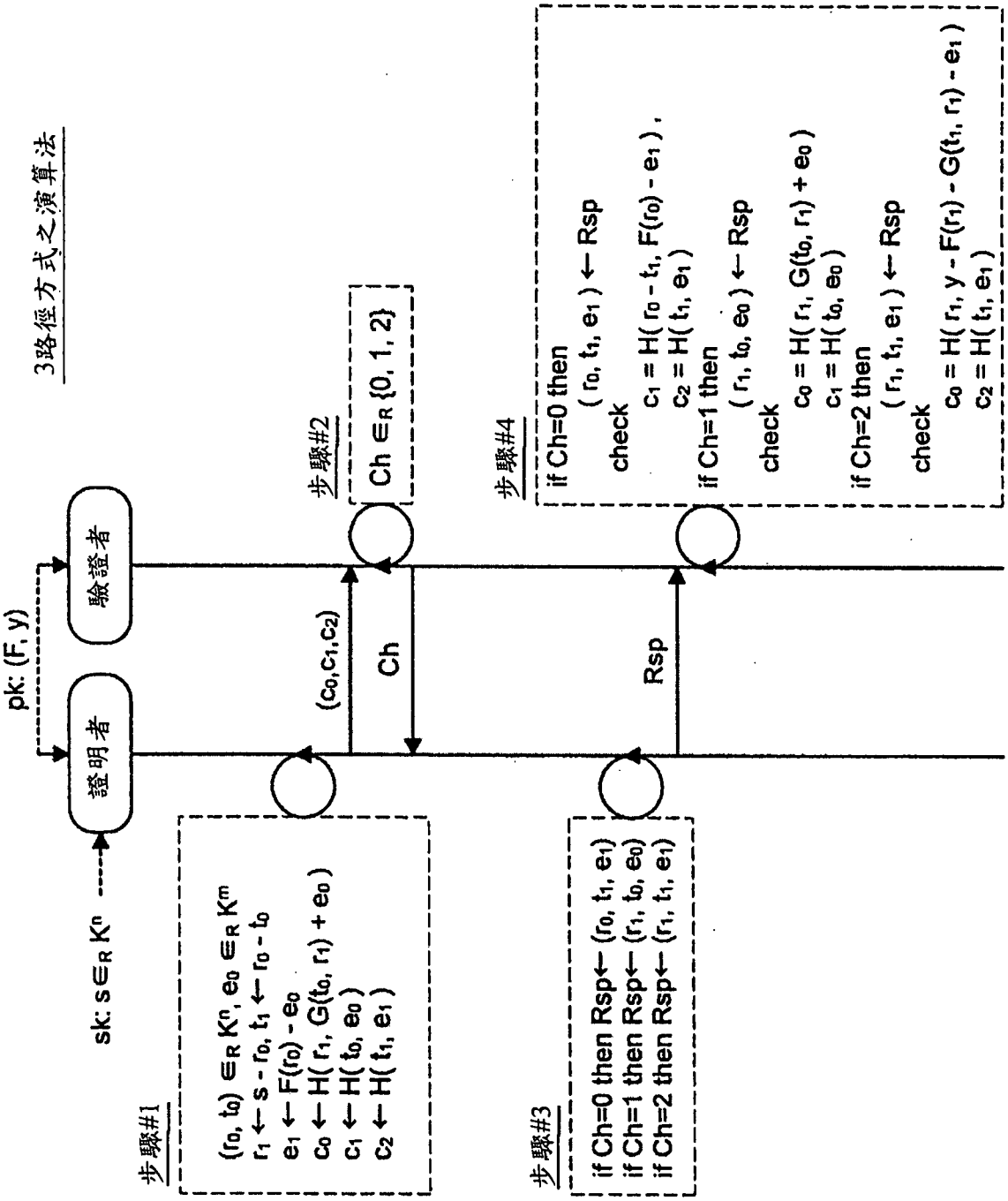

3路徑方式之演算法
證明者
驗證者
pk: (F, y)
sk: s∈R Kn
步驟#1
(r0, t0) ∈R Kn, e0 ∈R Km
r1 ← s - r0, t1 ← r0 - t0
e1 ← F(r0) - e0
c0 ← H(r1, G(t0, r1) + e0)
c1 ← H(t0, e0)
c2 ← H(t1, e1)
(c0,c1,c2)
步驟#2
Ch ∈R {0, 1, 2}
Ch
步驟#3
if Ch=0 then Rsp← (r0, t1, e1)
if Ch=1 then Rsp← (r1, t0, e0)
if Ch=2 then Rsp← (r1, t1, e1)
Rsp
步驟#4
if Ch=0 then
(r0, t1, e1) ← Rsp
check
c1 = H(r0 - t1, F(r0) - e1) ,
c2 = H(t1, e1)
if Ch=1 then
(r1, t0, e0) ← Rsp
check
c0 = H(r1, G(t0, r1) + e0)
c1 = H(t0, e0)
if Ch=2 then
(r1, t1, e1) ← Rsp
check
c0 = H(r1, y - F(r1) - G(t1, r1) - e1)
c2 = H(t1, e1)

圖 4

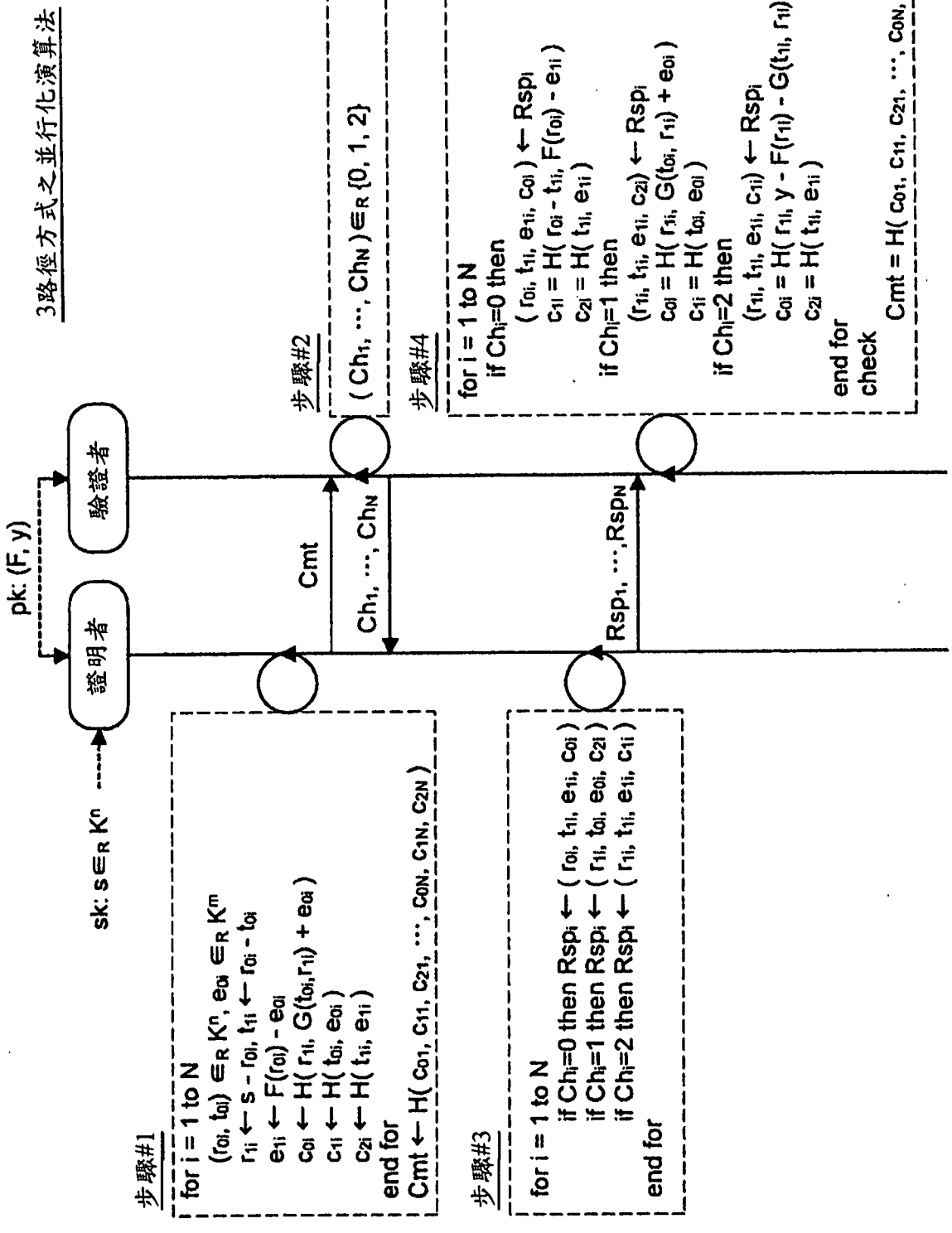

3路徑方式之並行化演算法
pk: (F, y)
sk: s∈R Kn
證明者
驗證者
Cmt
Ch1, ···, ChN
Rsp1, ···, RspN
步驟#1
for i = 1 to N
(r0i, t0i) ∈R Kn, e0i ∈R Km
r1i ← s - r0i, t1i ← r0i - t0i
e1i ← F(r0i) - e0i
c0i ← H(r1i, G(t0i,r1i) + e0i)
c1i ← H(t0i, e0i)
c2i ← H(t1i, e1i)
end for
Cmt ← H(c01, c11, c21, ···, c0N, c1N, c2N)
步驟#2
(Ch1, ···, ChN)∈R {0, 1, 2}
步驟#3
for i = 1 to N
if Chi=0 then Rspi ← (r0i, t1i, e1i, c0i)
if Chi=1 then Rspi ← (r1i, t0i, e0i, c2i)
if Chi=2 then Rspi ← (r1i, t1i, e1i, c1i)
end for
步驟#4
for i = 1 to N
if Chi=0 then
(r0i, t1i, e1i, c0i) ← Rspi
c1i = H(r0i - t1i, F(r0i) - e1i)
c2i = H(t1i, e1i)
if Chi=1 then
(r1i, t1i, e1i, c2i) ← Rspi
c0i = H(r1i, G(t0i, r1i) + e0i)
c1i = H(t0i, e0i)
if Chi=2 then
(r1i, t1i, e1i, c1i) ← Rspi
c0i = H(r1i, y - F(r1i) - G(t1i, r1i) - e1i)
c2i = H(t1i, e1i)
end for
check
Cmt = H(c01, c11, c21, ···, c0N, c1N, c2N)

圖5

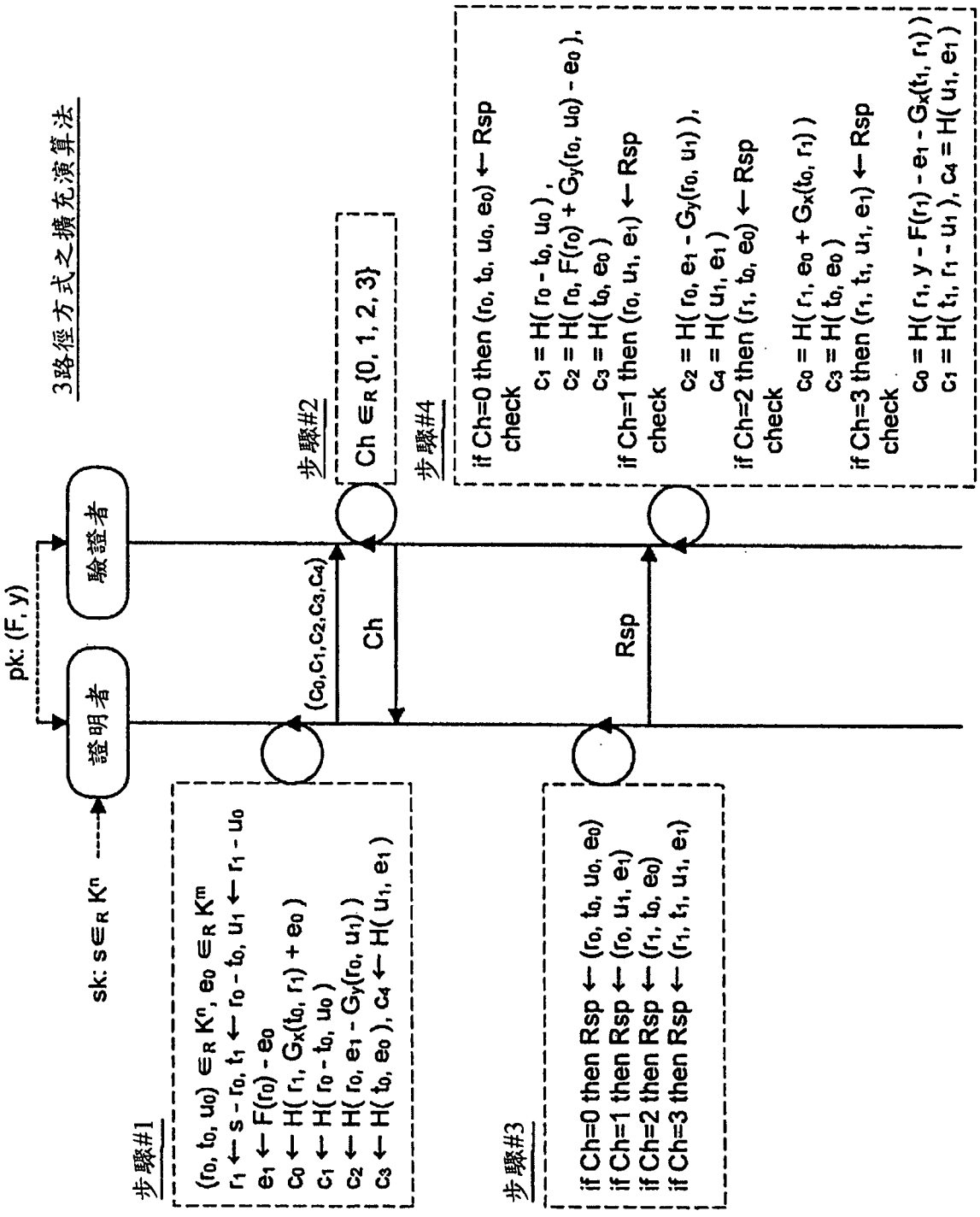

3路徑方式之擴充演算法
pk: (F, y)
sk: s∈R Kn
證明者
驗證者
步驟#1
(r0, t0, u0) ∈R Kn, e0 ∈R Km
r1 ← s - r0, t1 ← r0 - t0, u1 ← r1 - u0
e1 ← F(r0) - e0
c0 ← H(r1, Gx(t0, r1) + e0)
c1 ← H(r0 - t0, u0)
c2 ← H(r0, e1 - Gy(r0, u1))
c3 ← H(t0, e0), c4 ← H(u1, e1)
(c0,c1,c2,c3,c4)
Ch
步驟#2
Ch ∈R {0, 1, 2, 3}
步驟#3
if Ch=0 then Rsp ← (r0, t0, u0, e0)
if Ch=1 then Rsp ← (r0, u1, e1)
if Ch=2 then Rsp ← (r1, t0, e0)
if Ch=3 then Rsp ← (r1, t1, u1, e1)
Rsp
步驟#4
if Ch=0 then (r0, t0, u0, e0) ← Rsp
check
c1 = H(r0 - t0, u0),
c2 = H(r0, F(r0) + Gy(r0, u0) - e0),
c3 = H(t0, e0)
if Ch=1 then (r0, u1, e1) ← Rsp
check
c2 = H(r0, e1 - Gy(r0, u1)),
c4 = H(u1, e1)
if Ch=2 then (r1, t0, e0) ← Rsp
check
c0 = H(r1, e0 + Gx(t0, r1))
c3 = H(t0, e0)
if Ch=3 then (r1, t1, u1, e1) ← Rsp
check
c0 = H(r1, y - F(r1) - e1 - Gx(t1, r1))
c1 = H(t1, r1 - u1), c4 = H(u1, e1)

圖6

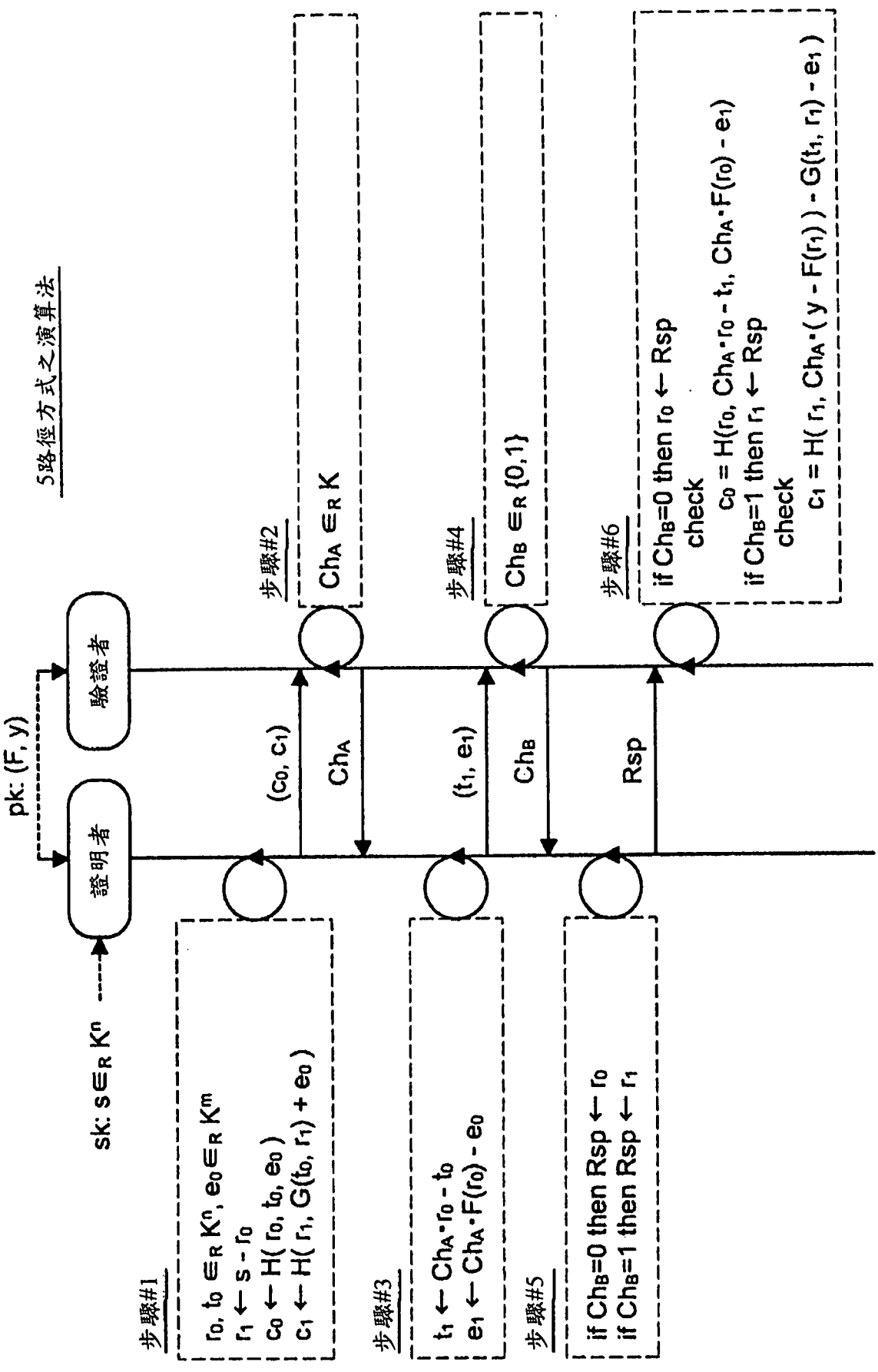

5路徑方式之演算法
證明者
驗證者
pk: (F, y)
sk: s∈R Kn
步驟#1
r0, t0 ∈R Kn, e0∈R Km
r1 ← s - r0
c0 ← H(r0, t0, e0)
c1 ← H(r1, G(t0, r1) + e0)
(c0, c1)
步驟#2
ChA ∈R K
ChA
步驟#3
t1 ← ChA·r0 - t0
e1 ← ChA·F(r0) - e0
(t1, e1)
步驟#4
ChB ∈R {0,1}
ChB
步驟#5
if ChB=0 then Rsp ← r0
if ChB=1 then Rsp ← r1
Rsp
步驟#6
if ChB=0 then r0 ← Rsp
check
c0 = H(r0, ChA·r0 - t1, ChA·F(r0) - e1)
if ChB=1 then r1 ← Rsp
check
c1 = H(r1, ChA·(y - F(r1)) - G(t1, r1) - e1)

圖 7

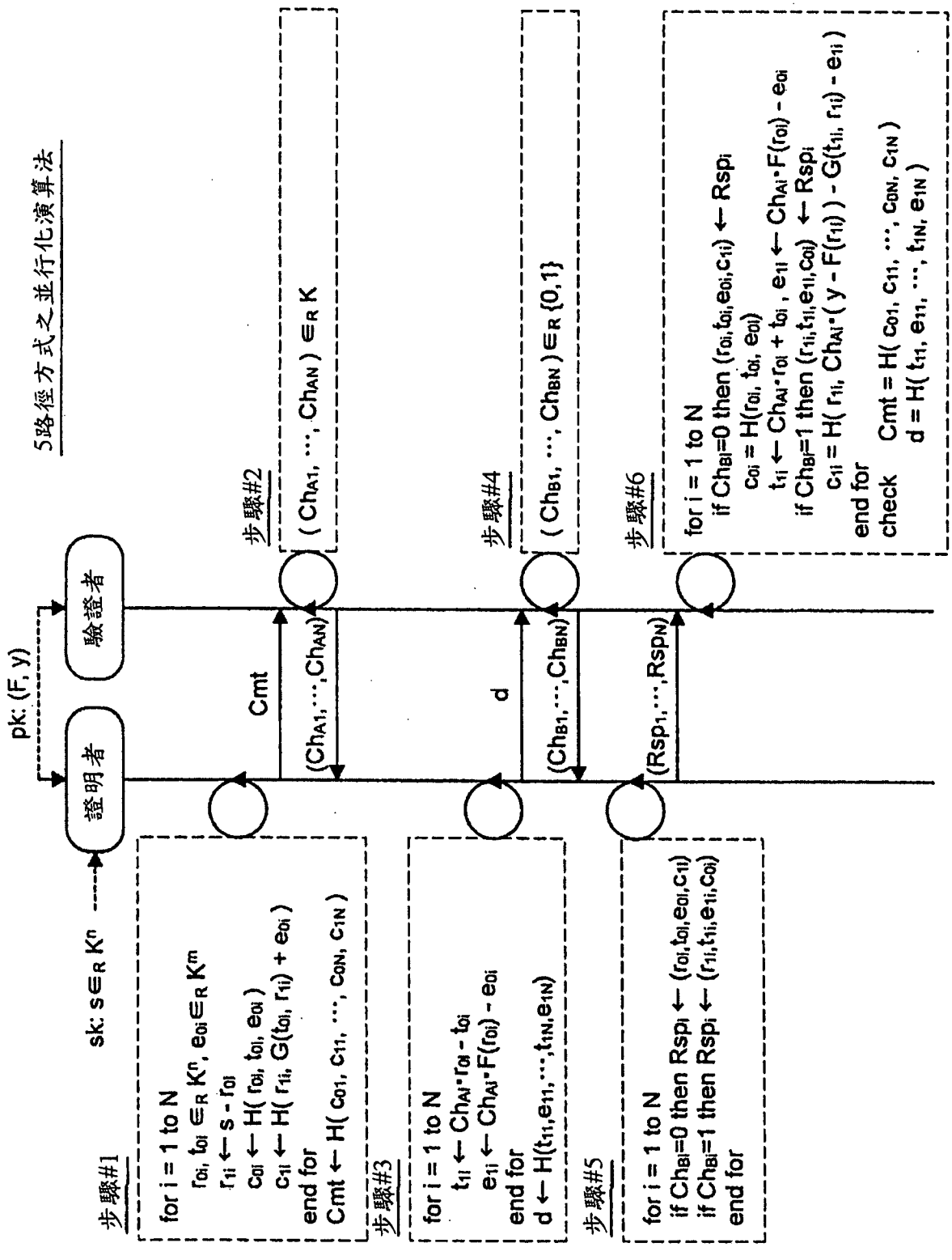

5路徑方式之並行化演算法
pk: (F, y)
sk: s∈R Kn
證明者
驗證者
步驟#1
for i = 1 to N
roi, toi ∈R Kn, eoi∈R Km
r1i ← s - roi
coi ← H(roi, toi, eoi)
c1i ← H(r1i, G(toi, r1i) + eoi)
end for
Cmt ← H(co1, c11, ···, coN, c1N)
Cmt
步驟#2
(ChA1, ···, ChAN) ∈R K
(ChA1,···,ChAN)
步驟#3
for i = 1 to N
t1i ← ChAi·roi - toi
e1i ← ChAi·F(roi) - eoi
end for
d ← H(t11,e11,···,t1N,e1N)
d
步驟#4
(ChB1, ···, ChBN) ∈R {0,1}
(ChB1,···,ChBN)
步驟#5
for i = 1 to N
if ChBi=0 then Rspi ← (roi,toi,eoi,c1i)
if ChBi=1 then Rspi ← (r1i,t1i,e1i,coi)
end for
(Rsp1,···,RspN)
步驟#6
for i = 1 to N
if ChBi=0 then (roi,toi,eoi,c1i) ← Rspi
coi = H(roi, toi, eoi)
t1i ← ChAi·roi + toi , e1i ← ChAi·F(roi) - eoi
if ChBi=1 then (r1i,t1i,e1i,coi) ← Rspi
c1i = H(r1i, ChAi·(y - F(r1i)) - G(t1i, r1i) - e1i)
end for
check Cmt = H(co1, c11, ···, coN, c1N)
d = H(t11, e11, ···, t1N, e1N)

圖8

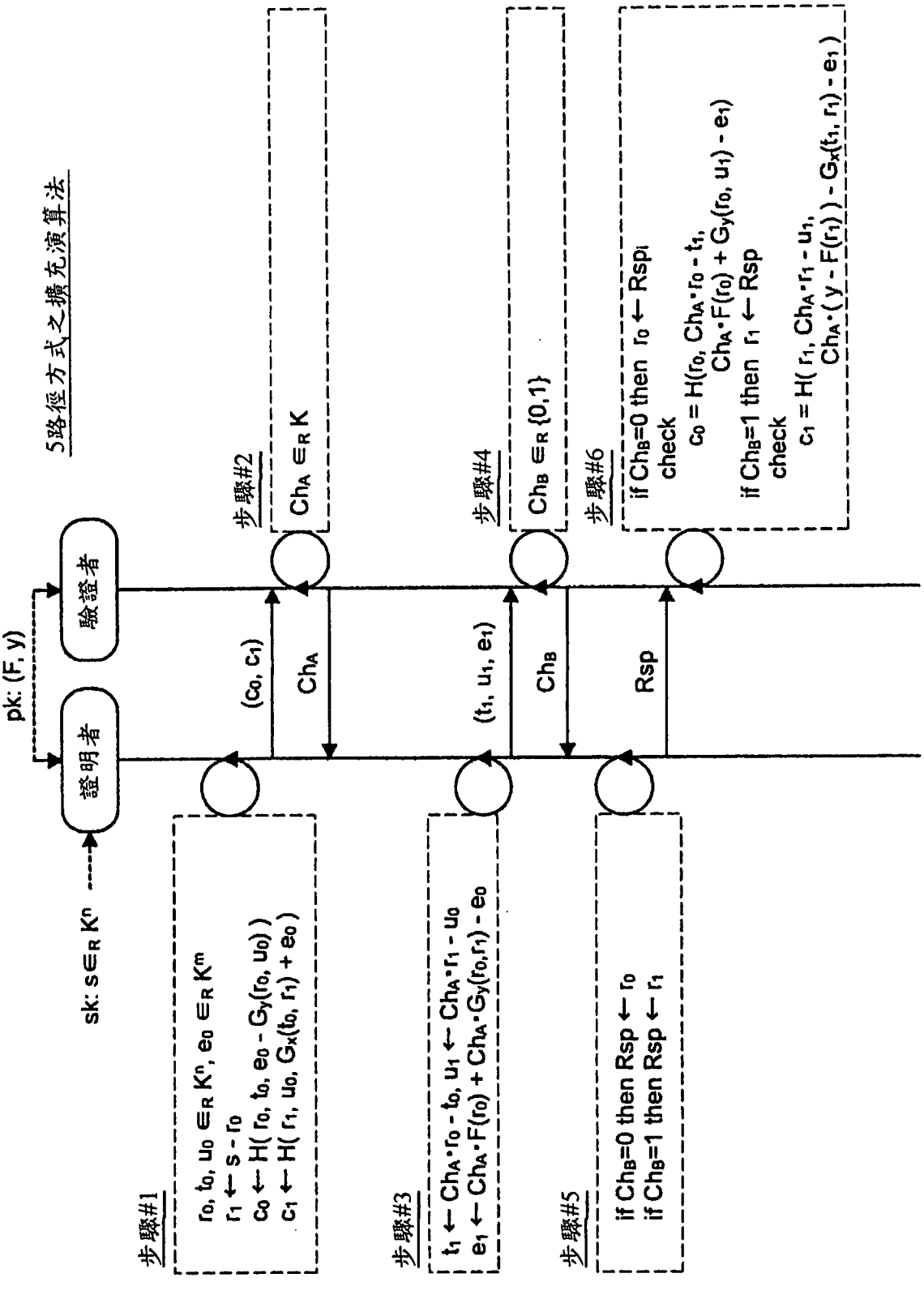

5路徑方式之擴充演算法
pk: (F, y)
sk: s∈R Kn
證明者
驗證者
步驟#1
r0, t0, u0 ∈R Kn, e0 ∈R Km
r1 ← s - r0
c0 ← H(r0, t0, e0 - Gy(r0, u0))
c1 ← H(r1, u0, Gx(t0, r1) + e0)
(c0, c1)
步驟#2
ChA ∈R K
ChA
步驟#3
t1 ← ChA·r0 - t0, u1 ← ChA·r1 - u0
e1 ← ChA·F(r0) + ChA·Gy(r0,r1) - e0
(t1, u1, e1)
步驟#4
ChB ∈R {0,1}
ChB
步驟#5
if ChB=0 then Rsp ← r0
if ChB=1 then Rsp ← r1
Rsp
步驟#6
if ChB=0 then r0 ← Rsp
check
c0 = H(r0, ChA·r0 - t1,
ChA·F(r0) + Gy(r0, u1) - e1)
if ChB=1 then r1 ← Rsp
check
c1 = H(r1, ChA·r1 - u1,
ChA·(y - F(r1)) - Gx(t1, r1) - e1)

圖9

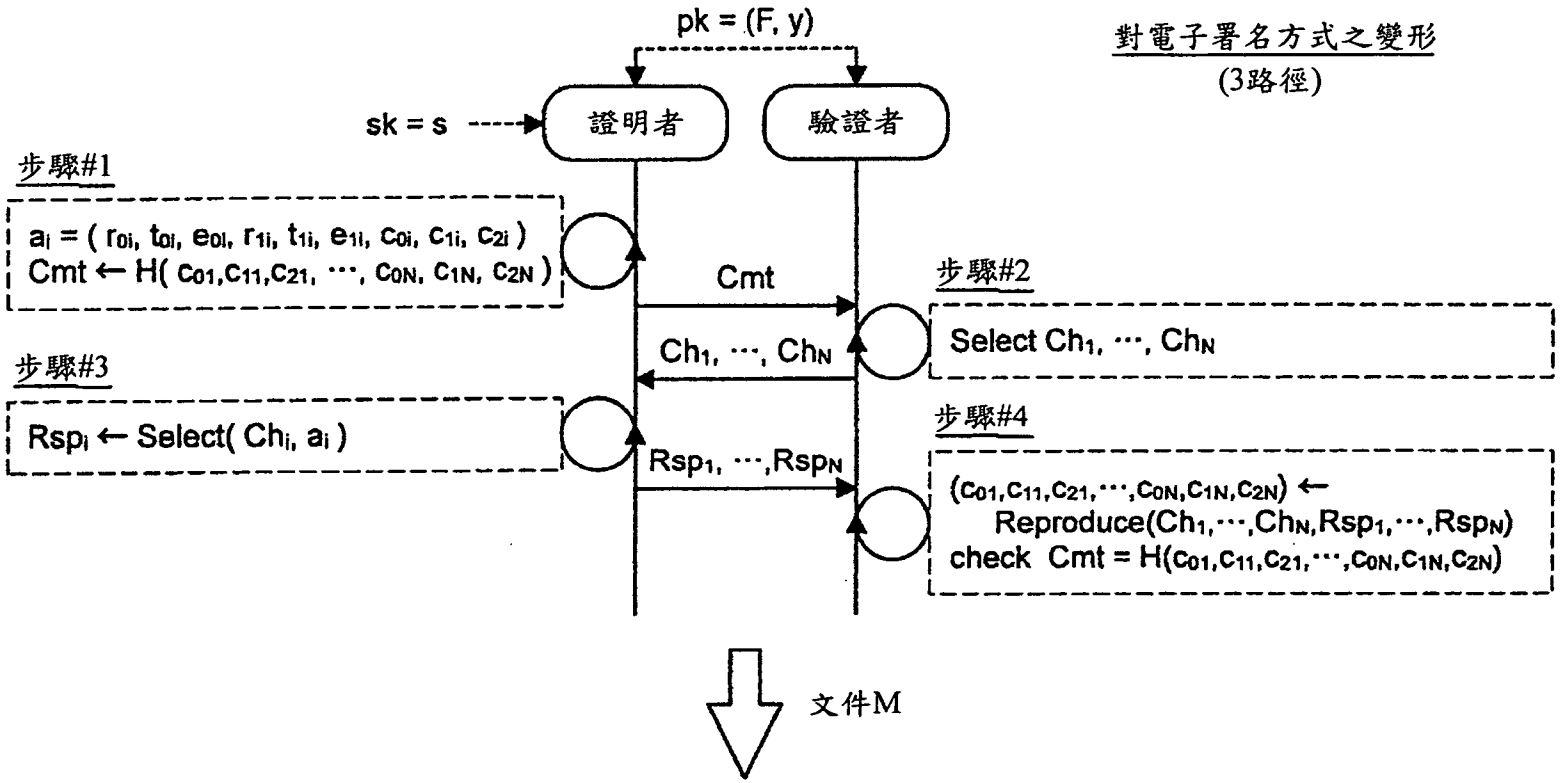

對電子署名方式之變形
(3路徑)
pk = (F, y)
sk = s
證明者
驗證者
步驟#1
$a_i = (r_{0i}, t_{0i}, e_{0i}, r_{1i}, t_{1i}, e_{1i}, c_{0i}, c_{1i}, c_{2i})$
$Cmt \leftarrow H(c_{01}, c_{11}, c_{21}, \cdots, c_{0N}, c_{1N}, c_{2N})$
Cmt
步驟#2
$Ch_1, \cdots, Ch_N$
Select $Ch_1, \cdots, Ch_N$
步驟#3
$Rsp_i \leftarrow Select(Ch_i, a_i)$
$Rsp_1, \cdots, Rsp_N$
步驟#4
$(c_{01}, c_{11}, c_{21}, \cdots, c_{0N}, c_{1N}, c_{2N}) \leftarrow$
Reproduce($Ch_1, \cdots, Ch_N, Rsp_1, \cdots, Rsp_N$)
check $Cmt = H(c_{01}, c_{11}, c_{21}, \cdots, c_{0N}, c_{1N}, c_{2N})$
文件M
署名生成演算法Sig
(1) $a_i = (r_{0i}, t_{0i}, e_{0i}, r_{1i}, t_{1i}, e_{1i}, c_{0i}, c_{1i}, c_{2i})$
(2) $Cmt \leftarrow H(c_{01}, c_{11}, c_{21}, \cdots, c_{0N}, c_{1N}, c_{2N})$
(3) $(Ch_1, \cdots, Ch_N) \leftarrow H(M, Cmt)$
(4) $Rsp_i \leftarrow Select(Ch_i, a_i)$
(5) 電子署名 : (Cmt, $Rsp_1, \cdots, Rsp_N$)
+
署名驗證演算法Ver
(1) $(Ch_1, \cdots, Ch_N) \leftarrow H(M, Cmt)$
(2) $(c_{01}, c_{11}, c_{21}, \cdots, c_{0N}, c_{1N}, c_{2N}) \leftarrow$
Reproduce($Ch_1, \cdots, Ch_N, Rsp_1, \cdots, Rsp_N$)
(3) check $Cmt = H(c_{01}, c_{11}, c_{21}, \cdots, c_{0N}, c_{1N}, c_{2N})$

圖10

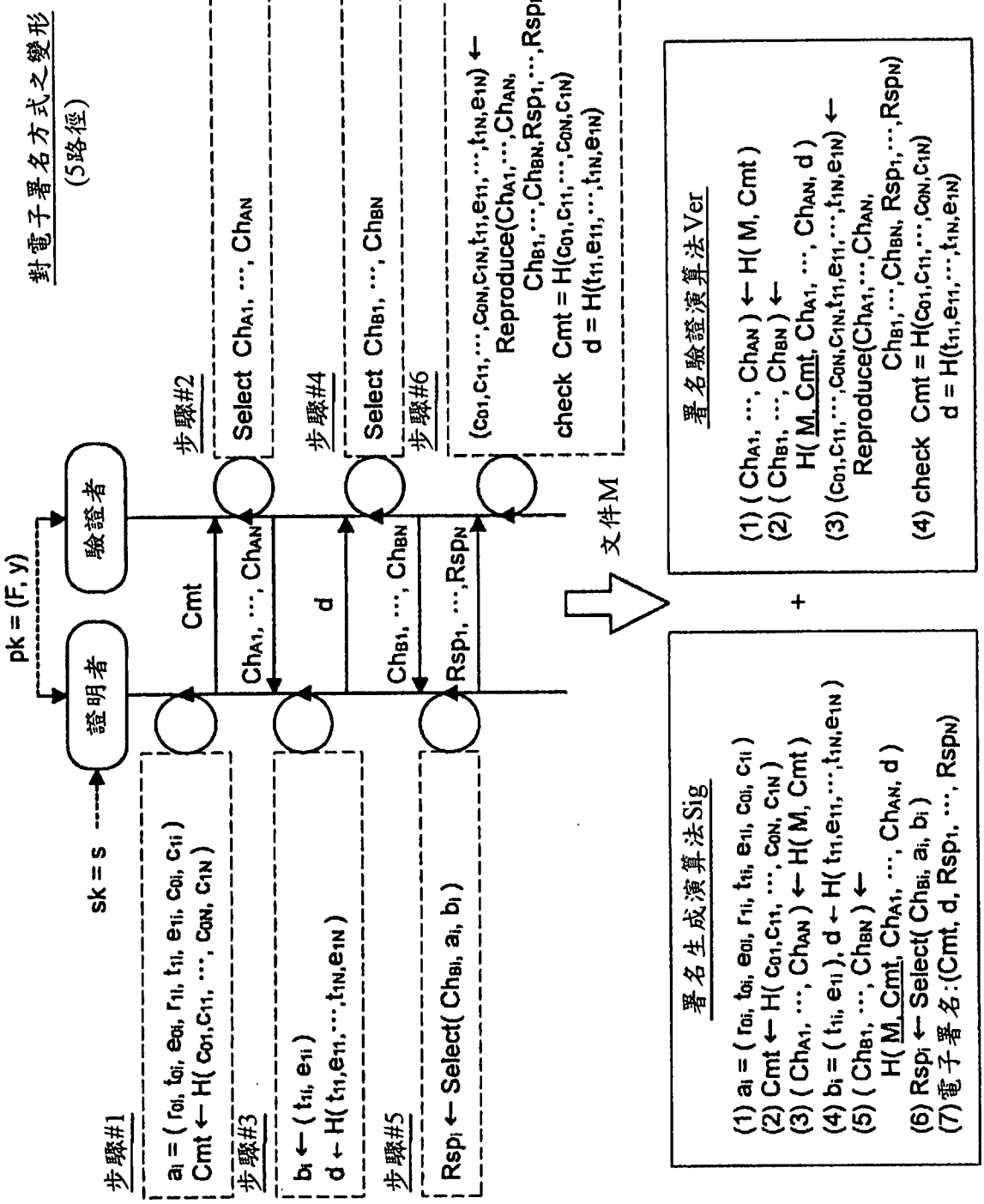
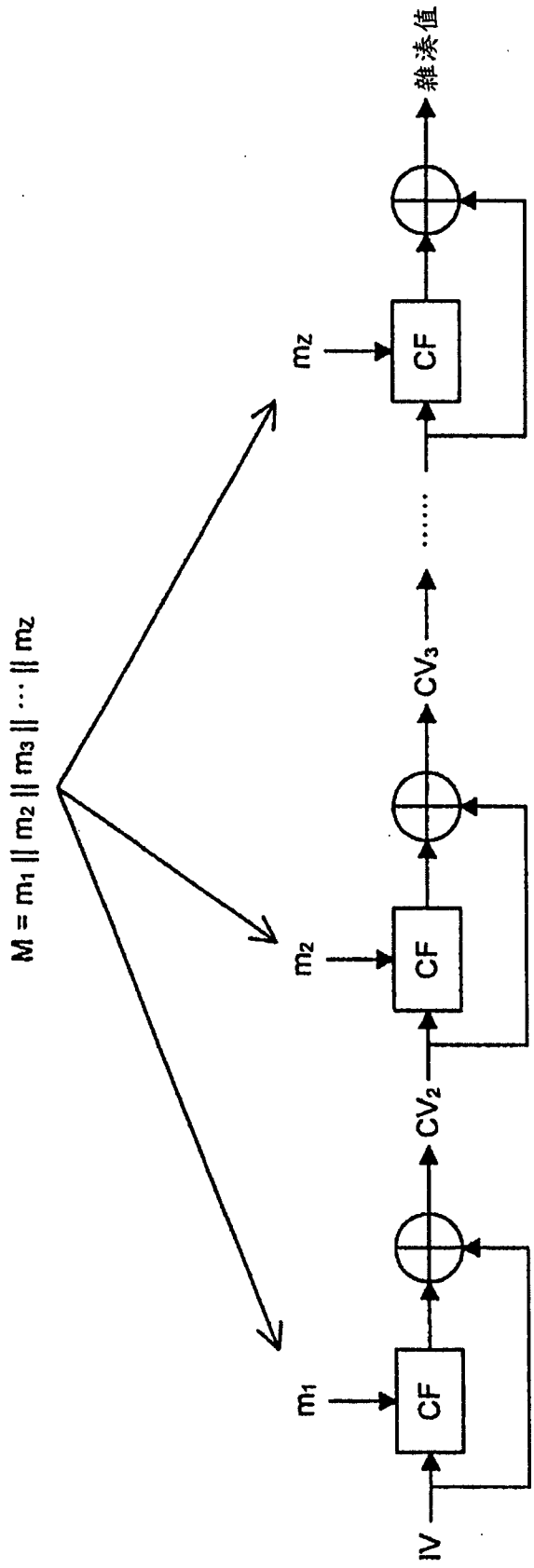

對電子署名方式之變形
(5路徑)
pk = (F, y)
sk = s
證明者
驗證者
Cmt
d
文件M
步驟#1
步驟#2
步驟#3
步驟#4
步驟#5
步驟#6
署名生成演算法Sig
署名驗證演算法Ver

圖11

雜湊函數之結構例



・IV：初始值
・CVi：鏈接值
・CF：壓縮函數

圖12

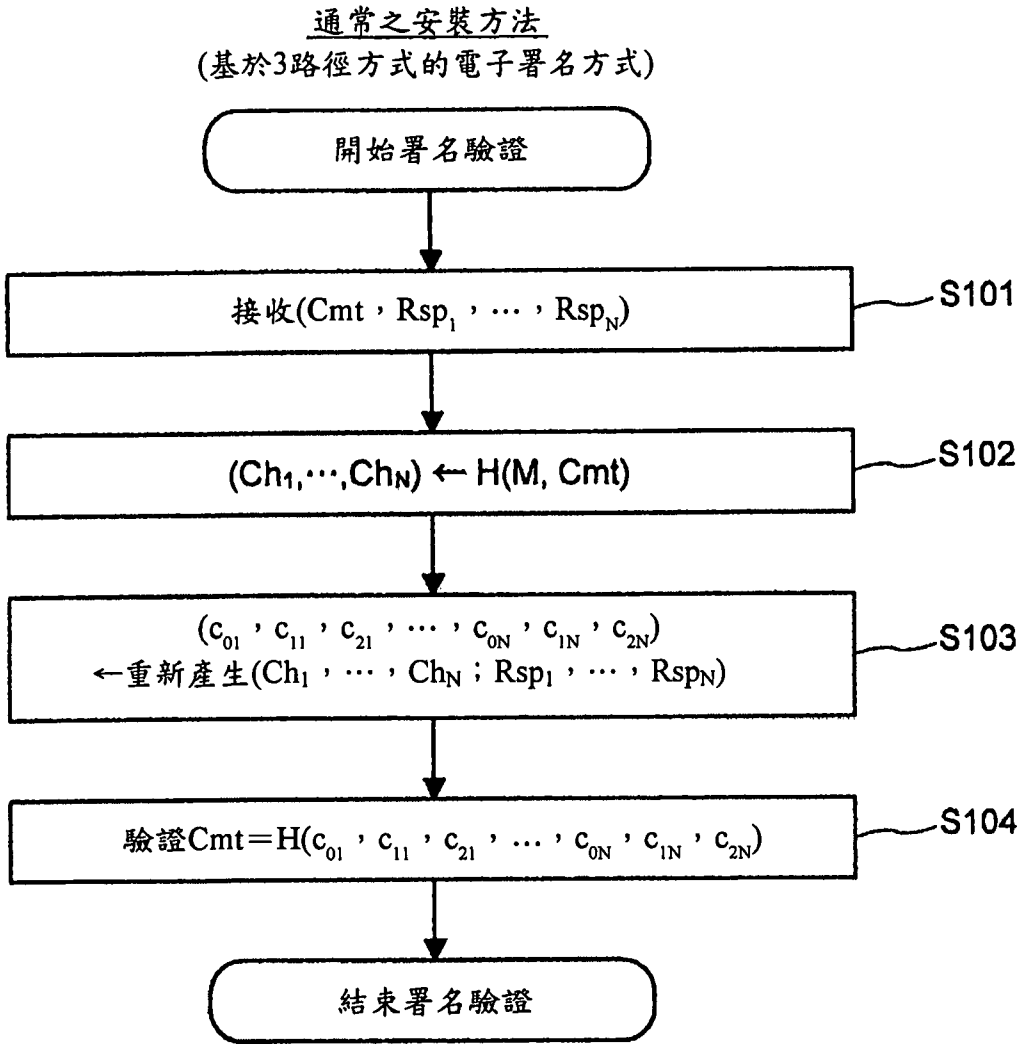

通常之安裝方法
(基於3路徑方式的電子署名方式)
開始署名驗證
接收(Cmt，Rsp_1，…，Rsp_N)
S101
$(Ch_1,\cdots,Ch_N) \leftarrow H(M, Cmt)$
S102
(c_{01}，c_{11}，c_{21}，…，c_{0N}，c_{1N}，c_{2N})
←重新產生(Ch_1，…，Ch_N；Rsp_1，…，Rsp_N)
S103
驗證$Cmt=H(c_{01}，c_{11}，c_{21}，\ldots，c_{0N}，c_{1N}，c_{2N})$
S104
結束署名驗證

圖13

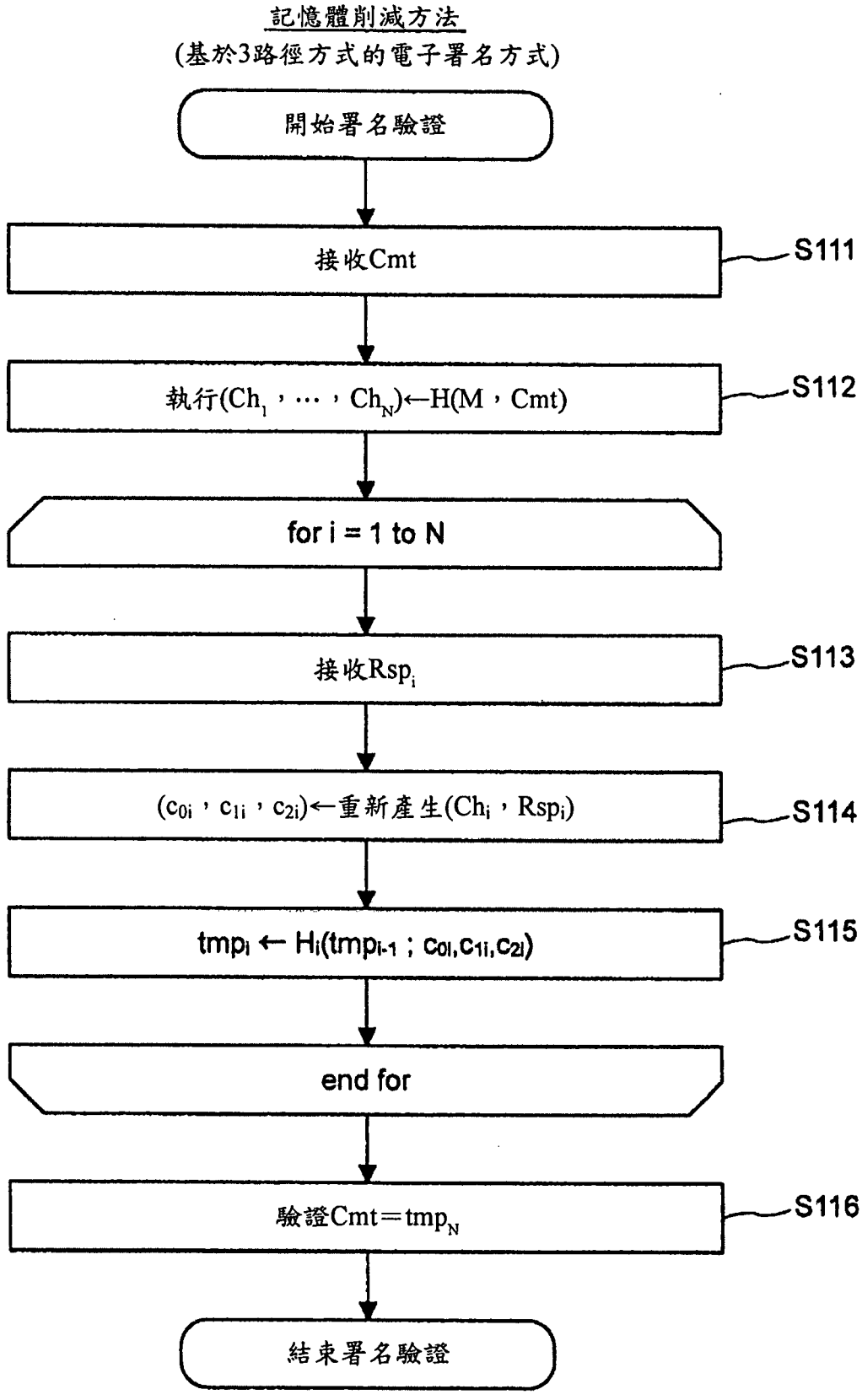

記憶體削減方法
(基於3路徑方式的電子署名方式)
開始署名驗證
接收Cmt
S111
執行(Ch1，…，ChN)←H(M，Cmt)
S112
for i = 1 to N
接收Rspi
S113
(c0i，c1i，c2i)←重新產生(Chi，Rspi)
S114
tmpi ← Hi(tmpi-1；c0i,c1i,c2i)
S115
end for
驗證Cmt＝tmpN
S116
結束署名驗證

圖14

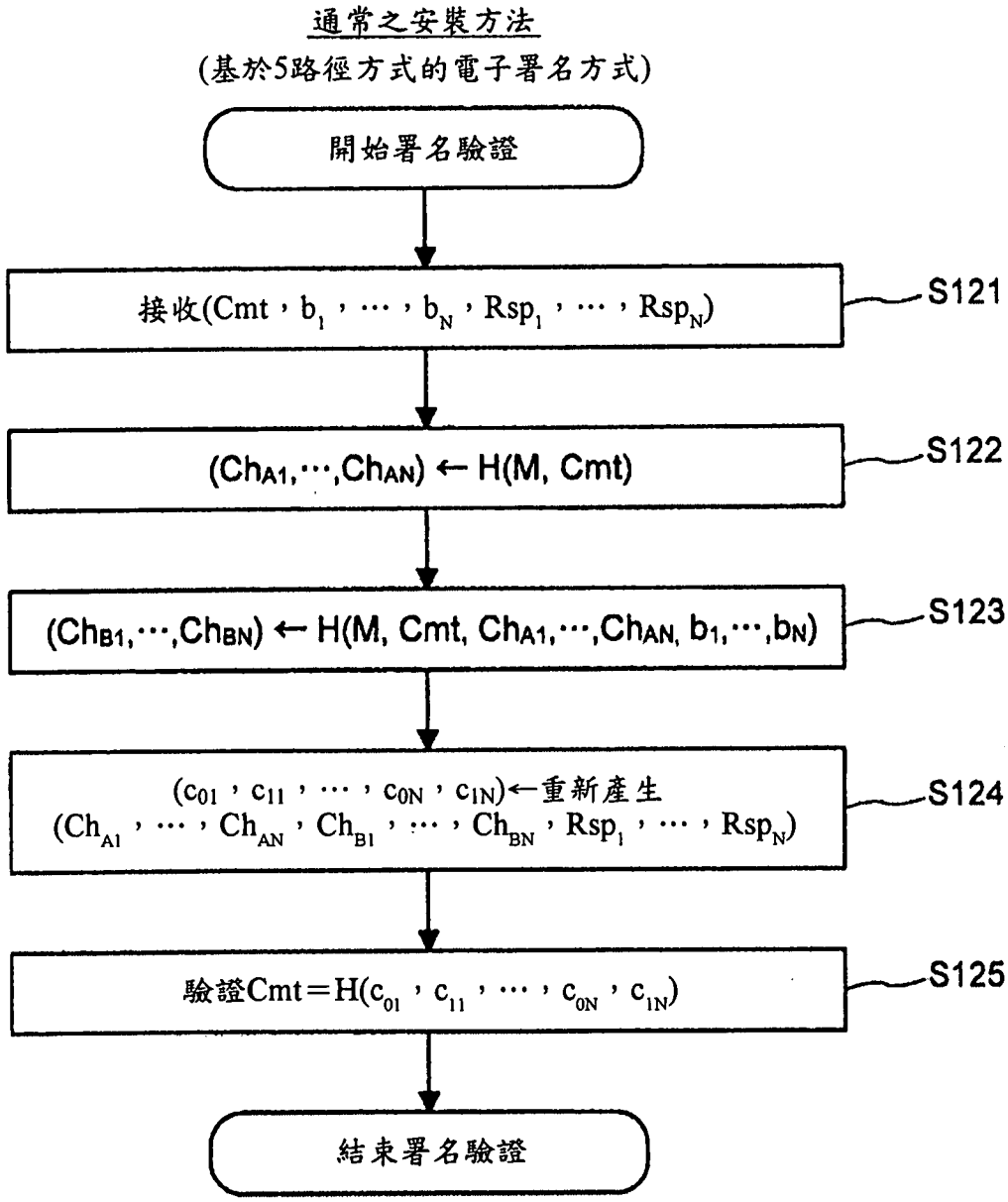

通常之安裝方法
(基於5路徑方式的電子署名方式)
開始署名驗證
接收(Cmt，b_1，…，b_N，Rsp_1，…，Rsp_N)
S121
$(Ch_{A1},\cdots,Ch_{AN}) \leftarrow H(M, Cmt)$
S122
$(Ch_{B1},\cdots,Ch_{BN}) \leftarrow H(M, Cmt, Ch_{A1},\cdots,Ch_{AN}, b_1,\cdots,b_N)$
S123
(c_{01}，c_{11}，…，c_{0N}，c_{1N})←重新產生
(Ch_{A1}，…，Ch_{AN}，Ch_{B1}，…，Ch_{BN}，Rsp_1，…，Rsp_N)
S124
驗證Cmt＝H(c_{01}，c_{11}，…，c_{0N}，c_{1N})
S125
結束署名驗證

圖15

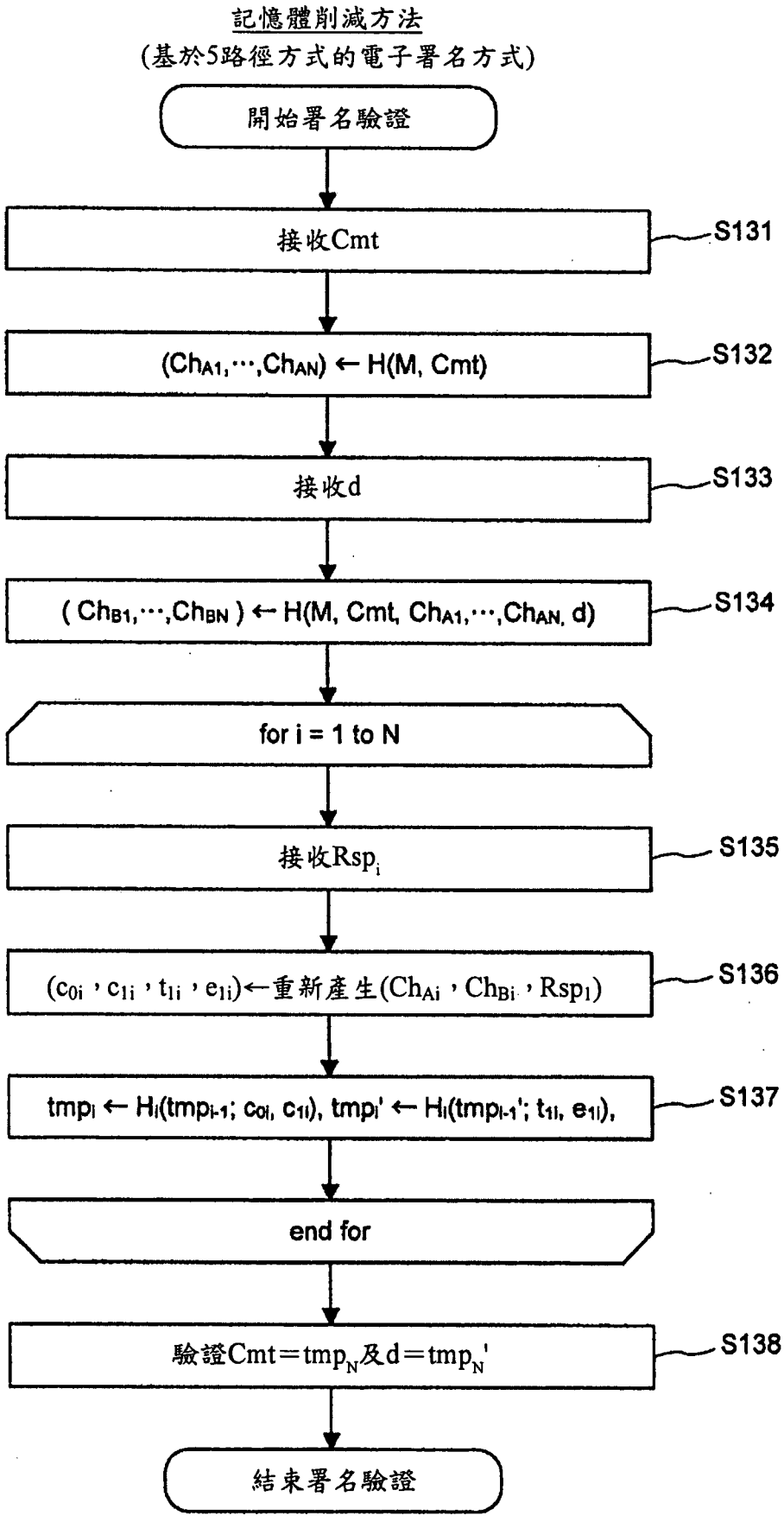
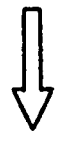

記憶體削減方法
(基於5路徑方式的電子署名方式)
開始署名驗證
接收Cmt
S131
(ChA1,⋯,ChAN) ← H(M, Cmt)
S132
接收d
S133
(ChB1,⋯,ChBN) ← H(M, Cmt, ChA1,⋯,ChAN, d)
S134
for i = 1 to N
接收Rsp_i
S135
(c0i , c1i , t1i , e1i)←重新產生(ChAi , ChBi , Rsp1)
S136
tmpl ← Hl(tmpl-1; c0l, c1l), tmpl' ← Hl(tmpl-1'; t1l, e1l),
S137
end for
驗證Cmt=tmpN及d=tmpN'
S138
結束署名驗證

圖16

抽選方法#1

(2進M符號之均勻隨機數)

X_{2M}： 0110110010101100 10 … 0101100100



(2位元地劃分：2進2符號→3進1符號)

01	10	11	00	10	10	11	00	10	…	01	01	10	01	00
‖	‖	‖	‖	‖	‖	‖	‖	‖		‖	‖	‖	‖	‖
X_{22}： 1	2	×	0	1	2	×	0	2		1	1	2	1	0

X_{22}＞2除外　　X_{22}＞2除外

圖17

抽選方法#2

(2進M符號之均勻隨機數)

X_{2M}： 0110110010101100 10 … 0101100100



(未劃分：2進2符號→3進1符號)

X_{2M}： 0110110010101100 10 … 0101100100

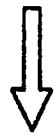
X_{2M}＞3^L除外

圖18

抽選方法#3

(2進M符號之均勻隨機數)

X_{2M}： 0110110010101100 10 … 0101100100



(k位元地劃分：2進2符號→3進1符號)

01101 | 10010 | 10110 | 010 … | 01011 | 00100

=

X_{2k}



$X_{2k} > 3^L$除外

圖19

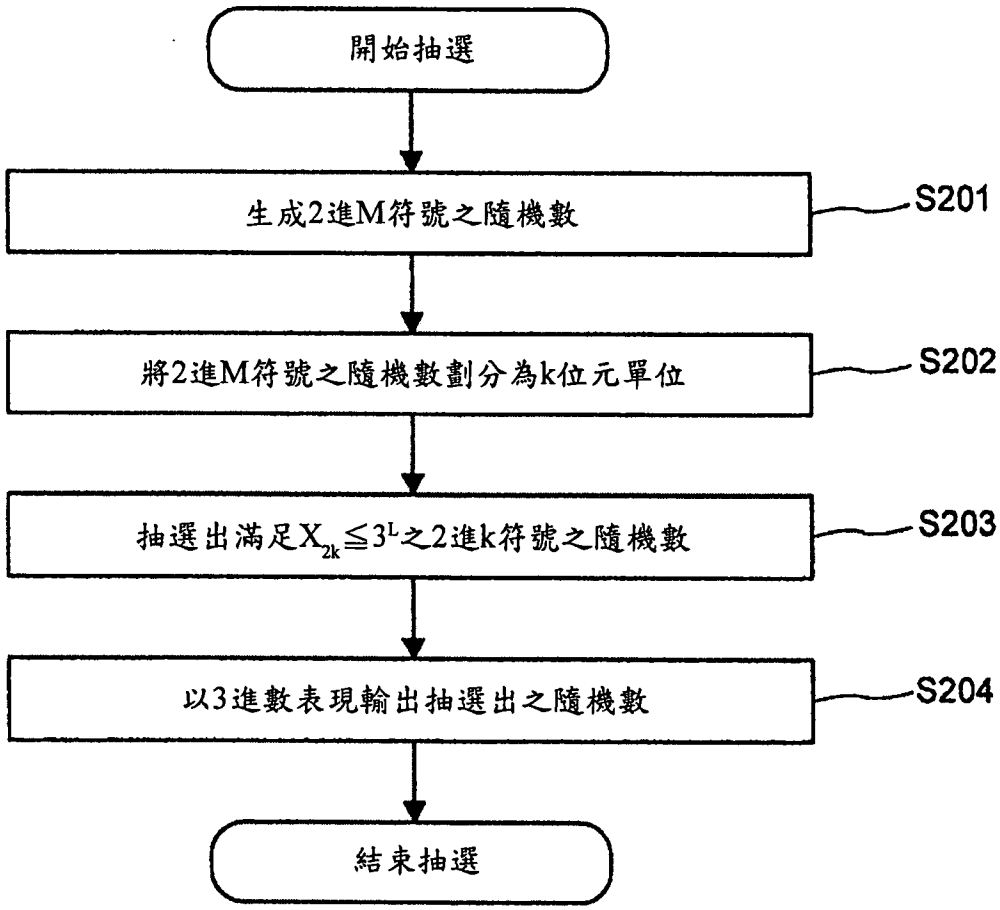


圖20

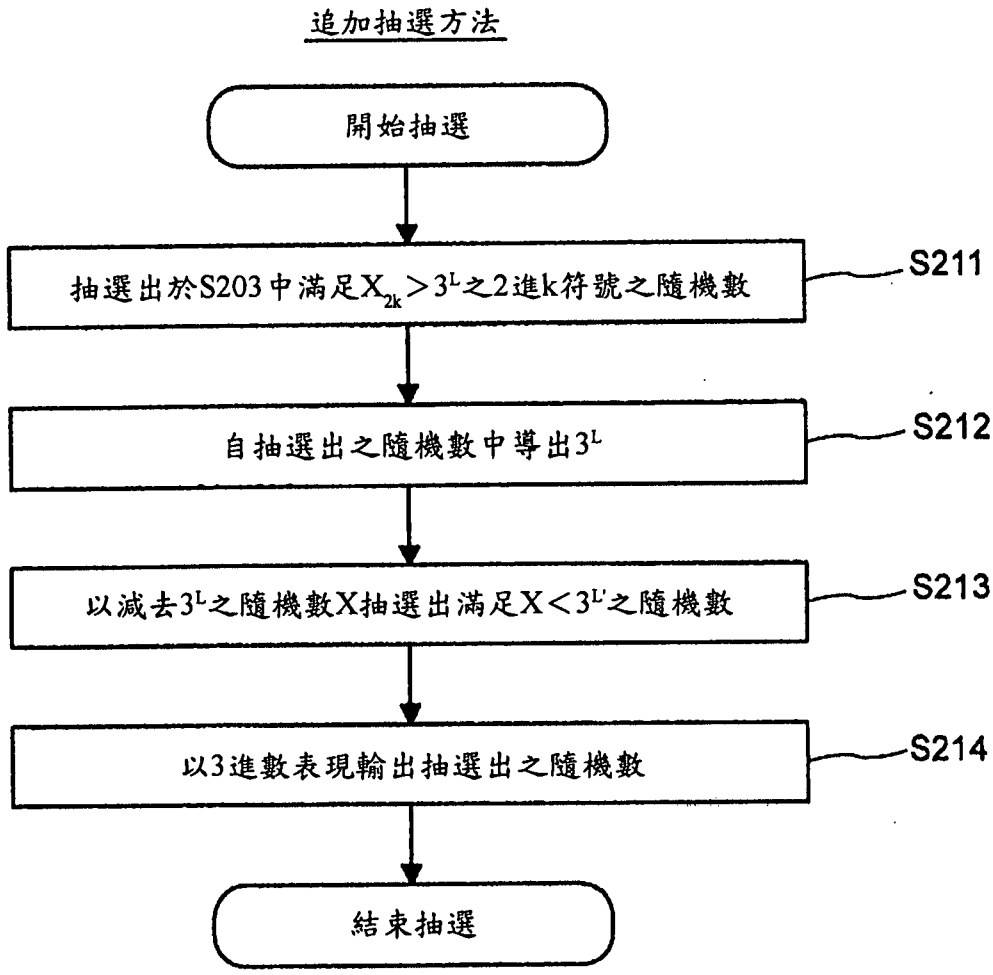

追加抽選方法
開始抽選
抽選出於S203中滿足$X_{2k}>3^L$之2進k符號之隨機數
S211
自抽選出之隨機數中導出3^L
S212
以減去3^L之隨機數X抽選出滿足$X<3^{L'}$之隨機數
S213
以3進數表現輸出抽選出之隨機數
S214
結束抽選

圖21

結構化方法#1(對F之應用)

$$F(x_1,\cdots,x_N)=\begin{cases}\sum_{I,J} a_{1IJ}x_I x_J+\sum_I b_{1I}x_I = y_1\\ \vdots\\ \sum_{I,J} a_{MIJ}x_I x_J+\sum_I b_{MI}x_I = y_M\end{cases}$$

資料結構A　　資料結構B

圖22

結構化方法#1(對G之應用)

$$G(x_1,\cdots,x_N,y_1,\cdots,y_N)=\begin{cases}\sum_{I,J}\left(a_{1IJ}+a_{1JI}\right)x_Iy_J=z_1\\ \vdots\\ \sum_{I,J}\left(a_{MIJ}+a_{MJI}\right)x_Iy_J=z_M\end{cases}$$

資料結構

圖23

結構化方法#1(對3次F之應用)

$$F(x_1,\cdots,x_N)=\begin{cases}\underbrace{\sum_{IJL} a_{1IJL}x_Ix_Jx_L}_{\text{資料結構A}}+\underbrace{\sum_{IJ} b_{1IJ}x_Ix_J}_{\text{資料結構B}}+\underbrace{\sum_{I} c_{1I}x_I}_{\text{資料結構C}}=y_1\\ \vdots\\ \underbrace{\sum_{IJL} a_{MIJL}x_Ix_Jx_L}_{\text{資料結構A}}+\underbrace{\sum_{IJ} b_{MIJ}x_Ix_J}_{\text{資料結構B}}+\underbrace{\sum_{I} c_{MI}x_I}_{\text{資料結構C}}=y_M\end{cases}$$

圖24

結構化方法#1(對3次G_x之應用)

$$G_x(x_1,\cdots,x_N,y_1,\cdots,y_N)=\begin{cases}\left[\sum_{IJL}\left(a_{IJL}+a_{ILJ}+a_{LJI}\right)y_Iy_Jx_L+\sum_{I,J}\left(b_{IJ}+b_{JI}\right)x_Iy_J=z_1\right] \\ \vdots \\ \left[\sum_{IJL}\left(a_{MJL}+a_{MLJ}+a_{LJM}\right)y_Iy_Jx_L+\sum_{IJ}\left(b_{MJ}+b_{MJI}\right)x_Iy_J=z_N\right]\end{cases}$$

資料結構A　　資料結構B

圖25

結構化方法#1(對3次G_y之應用)

$$G_y(x_1,\cdots,x_N,y_1,\cdots,y_N)=\begin{cases}\sum_{IJL}\left(a_{1IJL}+a_{1IIJ}+a_{1LJI}\right)x_Ix_Jy_L=z_1\\ \vdots\\ \sum_{IJL}\left(a_{MIJL}+a_{MIIJ}+a_{MLJI}\right)x_Ix_Jy_L=z_N\end{cases}$$

資料結構

圖26

結構化方法#2

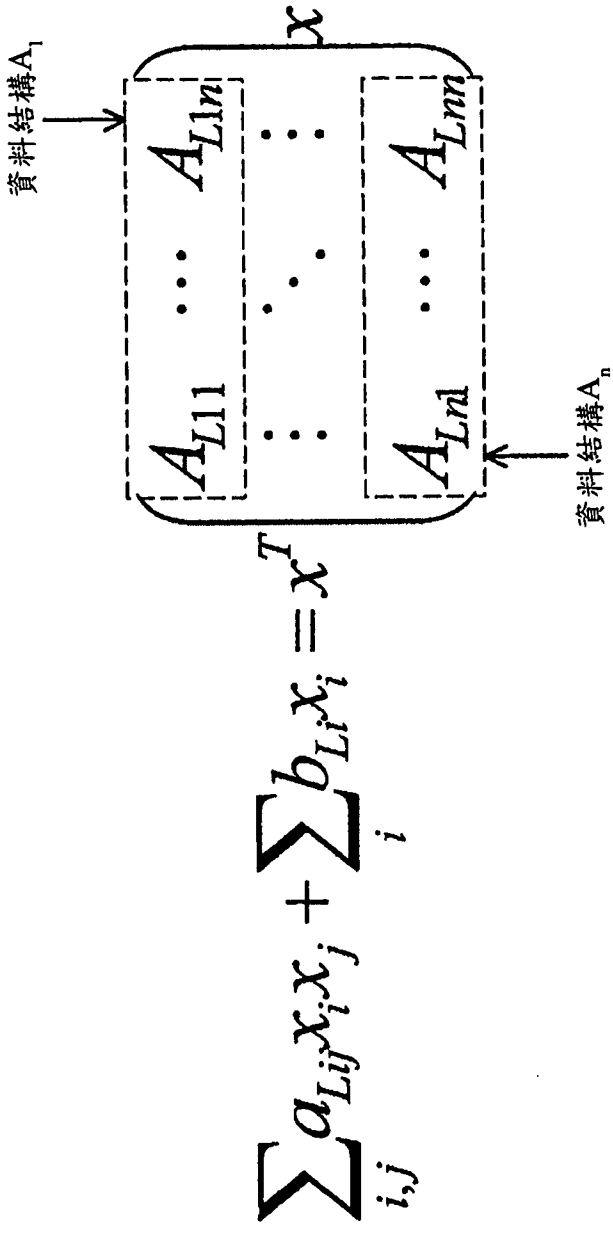

資料結構A1
資料結構An
$\sum_{i,j} a_{Lij} x_i x_j + \sum_i b_{Li} x_i = x^T \begin{pmatrix} A_{L11} & \cdots & A_{L1n} \\ \vdots & \ddots & \vdots \\ A_{Ln1} & \cdots & A_{Lnn} \end{pmatrix} x$

圖27

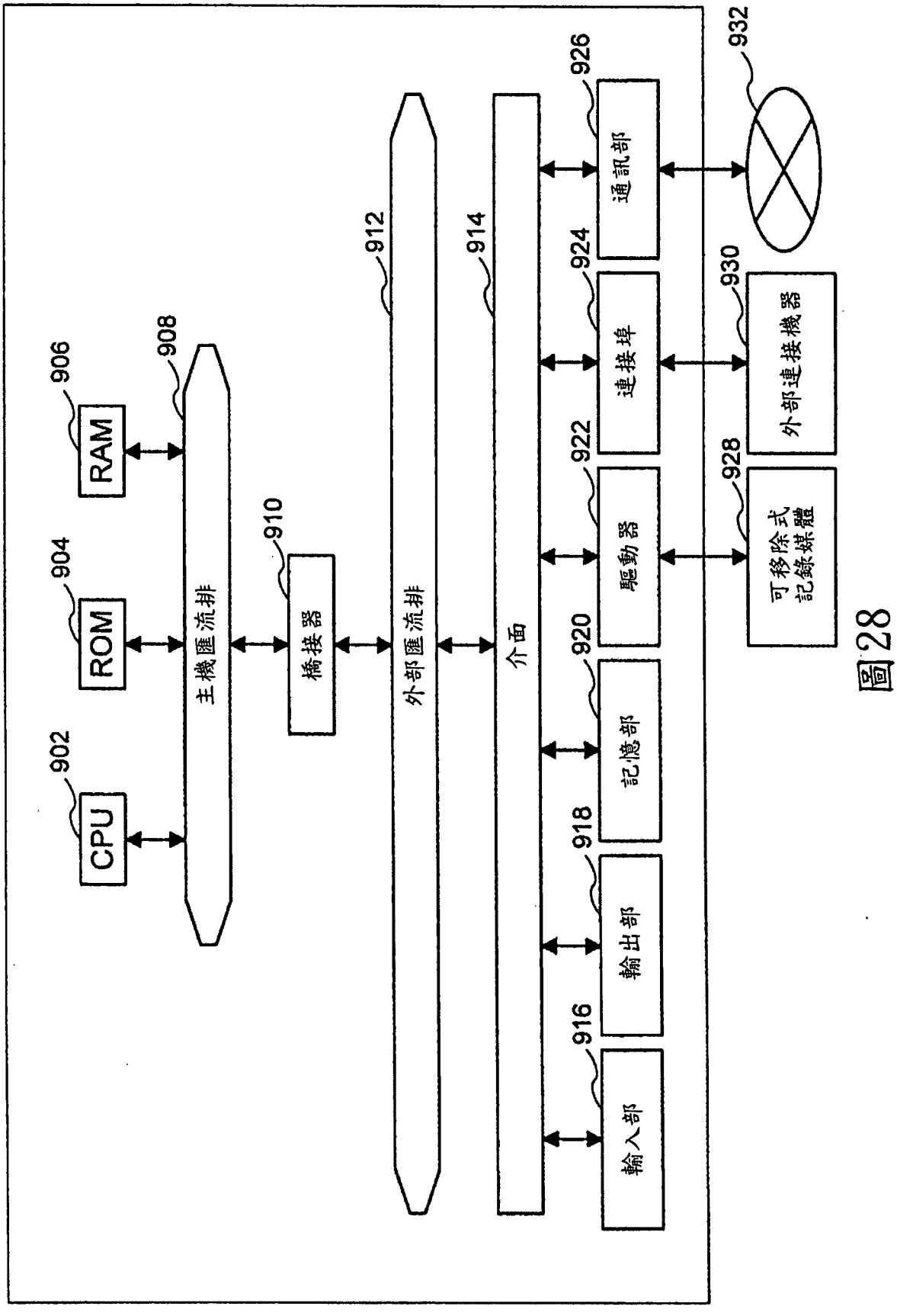

902
CPU
904
ROM
906
RAM
908
主機匯流排
910
橋接器
912
外部匯流排
914
介面
916
輸入部
918
輸出部
920
記憶部
922
驅動器
924
連接埠
926
通訊部
928
可移除式記錄媒體
930
外部連接機器
932

圖28