(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第5736816号
(P5736816)
(45)発行日　平成27年6月17日(2015.6.17)　(24)登録日　平成27年5月1日(2015.5.1)

(51)Int.Cl.　F I
H04L 9/32 (2006.01)　H04L 9/00 675C

請求項の数 23　(全 73 頁)

(21)出願番号　特願2011-26401(P2011-26401)
(22)出願日　平成23年2月9日(2011.2.9)
(65)公開番号　特開2012-98690(P2012-98690A)
(43)公開日　平成24年5月24日(2012.5.24)
審査請求日　平成26年2月10日(2014.2.10)
(31)優先権主張番号　特願2010-125026(P2010-125026)
(32)優先日　平成22年5月31日(2010.5.31)
(33)優先権主張国　日本国(JP)
(31)優先権主張番号　特願2010-224753(P2010-224753)
(32)優先日　平成22年10月4日(2010.10.4)
(33)優先権主張国　日本国(JP)

(73)特許権者　000002185
ソニー株式会社
東京都港区港南１丁目７番１号
(74)代理人　100095957
弁理士　亀谷　美明
(74)代理人　100096389
弁理士　金本　哲男
(74)代理人　100101557
弁理士　萩原　康司
(74)代理人　100128587
弁理士　松本　一騎
(72)発明者　作本　紘一
東京都港区港南１丁目７番１号　ソニー株式会社内

最終頁に続く

(54)【発明の名称】認証装置、認証方法、プログラム、及び署名生成装置

(57)【特許請求の範囲】
【請求項１】
ｓ∈Ｋ^{n}を秘密鍵に設定し、環Ｋ上の多次多項式ｆ$_{i}$（ｘ$_{1}$，…，ｘ$_{n}$）（ｉ＝１～ｍ）を公開鍵またはシステムパラメータ、及びｙ$_{i}$＝ｆ$_{i}$（ｓ）を公開鍵に設定する鍵設定部と、
検証装置に対してメッセージｃを送信するメッセージ送信部と、
１つの前記メッセージｃに対するｋ通り（ｋ≧３）の検証パターンの中から前記検証装置により選択された１つの検証パターンの情報を受信する検証パターン受信部と、
ｋ通りの回答情報の中から、前記検証パターン受信部により受信された検証パターンの情報に対応する回答情報を前記検証装置に送信する回答送信部と、
を備え、
前記回答情報は、前記ｋ通りの回答情報を用いて実施した前記メッセージｃに対するｋ通りの検証パターンが全て成功した場合に秘密鍵ｓが計算可能となる情報である、
認証装置。
【請求項２】
前記メッセージ送信部により単数又は複数のメッセージｃを送信する第１ステップ、それぞれのメッセージｃに対して前記検証パターン受信部により前記検証装置からそれぞれ検証パターンの情報を受信する第２ステップ、それぞれの検証パターンの情報に対して前記回答送信部により回答情報を送信する第３ステップを実行し、前記検証装置により全ての回答情報で検証が成功した場合に認証が成功する、

請求項１に記載の認証装置。
【請求項３】
前記メッセージ送信部により単数又は複数のメッセージｃを送信する第１ステップ、それぞれのメッセージｃに対して前記検証パターン受信部により前記検証装置からそれぞれ検証パターンの情報を受信する第２ステップ、それぞれの検証パターンの情報に対して前記回答送信部により回答情報を送信する第３ステップを実行する処理を繰り返し、
前記第１～第３ステップを所定回数実行した結果、前記検証装置により毎回全ての回答情報で検証が成功した場合に認証が成功する、
請求項２に記載の認証装置。
【請求項４】
前記メッセージｃがｃ＝（c_1，…，c_m）である場合、
前記メッセージ送信部は、一方向性関数Ｈを用いて新たなメッセージｃ’＝Ｈ（ｃ）を算出して、前記検証装置に対してメッセージｃ’を送信し、
前記回答送信部は、前記回答情報と共に、当該回答情報を利用しても前記検証装置が復元することができないメッセージｃの要素を送信する、
請求項２又は３に記載の認証装置。
【請求項５】
$s \in K^n$が秘密鍵に設定され、環Ｋ上の多次多項式f_i（x_1，…，x_n）（ｉ＝１～ｍ）が公開鍵またはシステムパラメータ、及び$y_i = f_i$（ｓ）が公開鍵に設定されており、
証明装置からメッセージｃを受信するメッセージ受信部と、
１つの前記メッセージｃに対するｋ通り（ｋ≧３）の検証パターンの中から１つの検証パターンを選択する検証パターン選択部と、
前記検証パターン選択部により選択された検証パターンの情報を前記証明装置に対して送信する検証パターン送信部と、
ｋ通りの回答情報の中から、前記検証パターン送信部により送信された検証パターンの情報に対応する回答情報を前記証明装置から受信する回答受信部と、
前記メッセージ受信部により受信されたメッセージｃ、及び前記回答受信部により受信された回答情報を用いて前記証明装置の正当性を検証する検証部と、
を備え、
前記回答情報は、前記ｋ通りの回答情報を用いて実施した前記メッセージｃに対するｋ通りの検証パターンが全て成功した場合に秘密鍵ｓが計算可能となる情報である、
認証装置。
【請求項６】
$s \in K^n$を秘密鍵に設定し、環Ｋ上の多次多項式f_i（x_1，…，x_n）（ｉ＝１～ｍ）を公開鍵またはシステムパラメータ、及び$y_i = f_i$（ｓ）を公開鍵に設定する鍵設定部と、
検証装置に対してメッセージｃを送信するメッセージ送信部と、
前記検証装置から応答αを受信する応答受信部と、
前記応答受信部により受信された応答αを用いて、前記メッセージｃに対する検証に利用される多項式ｆ”を生成する多項式生成部と、
前記多項式生成部により生成された多項式ｆ”を前記検証装置に送信する多項式送信部と、
１つの前記メッセージｃに対するｋ通り（ｋ≧２）の検証パターンの中から前記検証装置により選択された１つの検証パターンの情報を受信する検証パターン受信部と、
ｋ通りの回答情報の中から、前記検証パターン受信部により受信された検証パターンの情報に対応する回答情報を前記検証装置に送信する回答送信部と、
を備え、
前記回答情報は、２通りの前記応答α、前記多項式ｆ”及び前記ｋ通りの回答情報を用いて実施した前記メッセージｃに対する計２ｋ通りの応答と検証パターンの組合せが全て

成功した場合に秘密鍵ｓが計算可能となる情報である、
認証装置。
【請求項７】
前記メッセージ送信部により単数又は複数のメッセージｃを送信する第１ステップ、それぞれのメッセージｃに対して前記応答受信部から応答αを受信する第２ステップ、当該第２ステップで受信した応答αを用いて前記多項式生成部によりそれぞれ多項式ｆ”を生成し、前記多項式送信部により当該多項式ｆ”を送信する第３ステップ、それぞれのメッセージｃに対して前記検証パターン受信部により前記検証装置からそれぞれ検証パターンの情報を受信する第４ステップ、それぞれの検証パターンの情報に対して前記回答送信部により回答情報を送信する第５ステップを実行し、前記検証装置により全ての回答情報で検証が成功した場合に認証が成功する、
請求項６に記載の認証装置。
【請求項８】
前記メッセージ送信部により単数又は複数のメッセージｃを送信する第１ステップ、それぞれのメッセージｃに対して前記応答受信部から応答αを受信する第２ステップ、当該第２ステップで受信した応答αを用いて前記多項式生成部によりそれぞれ多項式ｆ”を生成し、前記多項式送信部により当該多項式ｆ”を送信する第３ステップ、それぞれのメッセージｃに対して前記検証パターン受信部により前記検証装置からそれぞれ検証パターンの情報を受信する第４ステップ、それぞれの検証パターンの情報に対して前記回答送信部により回答情報を送信する第５ステップを実行する処理を繰り返し、
前記第１～第５ステップを所定回数実行した結果、前記検証装置により毎回全ての回答情報で検証が成功した場合に認証が成功する、
請求項７に記載の認証装置。
【請求項９】
$s \in K^n$が秘密鍵に設定され、環Ｋ上の多次多項式$f_i(x_1, \cdots, x_n)$（ｉ＝１～ｍ）が公開鍵またはシステムパラメータ、及び$y_i = f_i(s)$が公開鍵に設定されており、
証明装置からメッセージｃを受信するメッセージ受信部と、
前記証明装置に対して応答αを送信する応答送信部と、
前記応答送信部により送信された応答αを用いて前記証明装置により生成された、前記メッセージｃに対する検証に利用される多項式ｆ”を受信する多項式受信部と、
１つの前記メッセージｃに対するｋ通り（ｋ≧２）の検証パターンの中から１つの検証パターンを選択する検証パターン選択部と、
前記検証パターン選択部により選択された検証パターンの情報を前記証明装置に対して送信する検証パターン送信部と、
ｋ通りの回答情報の中から、前記検証パターン送信部により送信された検証パターンの情報に対応する回答情報を前記証明装置から受信する回答受信部と、
前記メッセージ受信部により受信されたメッセージｃ、前記多項式受信部により受信された多項式ｆ”、及び前記回答受信部により受信された回答情報を用いて前記証明装置の正当性を検証する検証部と、
を備え、
前記回答情報は、２通りの前記応答α、前記多項式ｆ”及び前記ｋ通りの回答情報を用いて実施した前記メッセージｃに対する計２ｋ通りの応答と検証パターンの組合せが全て成功した場合に秘密鍵ｓが計算可能となる情報である、
認証装置。
【請求項１０】
前記ｍ及びｎは、ｍ＜ｎの関係を有する、
請求項１、５、６、９のいずれか１項に記載の認証装置。
【請求項１１】
前記ｍ及びｎは、$2^{m-n} \ll 1$の関係を有する、

請求項１０に記載の認証装置。
【請求項１２】
ｓ∈Ｋ^{n}を秘密鍵に設定し、環Ｋ上の多次多項式ｆ$_{i}$（ｘ$_{1}$，…，ｘ$_{n}$）（ｉ＝１～ｍ）を公開鍵またはシステムパラメータ、及びｙ$_{i}$＝ｆ$_{i}$（ｓ）を公開鍵に設定する鍵設定ステップと、
検証装置に対してメッセージｃを送信するメッセージ送信ステップと、
１つの前記メッセージｃに対するｋ通り（ｋ≧３）の検証パターンの中から前記検証装置により選択された１つの検証パターンの情報を受信する検証パターン受信ステップと、
ｋ通りの回答情報の中から、前記検証パターン受信ステップで受信された検証パターンの情報に対応する回答情報を前記検証装置に送信する回答送信ステップと、
を含み、
前記回答情報は、前記ｋ通りの回答情報を用いて実施した前記メッセージｃに対するｋ通りの検証パターンが全て成功した場合に秘密鍵ｓが計算可能となる情報である、
認証方法。
【請求項１３】
ｓ∈Ｋ^{n}が秘密鍵に設定され、環Ｋ上の多次多項式ｆ$_{i}$（ｘ$_{1}$，…，ｘ$_{n}$）（ｉ＝１～ｍ）が公開鍵またはシステムパラメータ、及びｙ$_{i}$＝ｆ$_{i}$（ｓ）が公開鍵に設定されており、
証明装置からメッセージｃを受信するメッセージ受信ステップと、
１つの前記メッセージｃに対するｋ通り（ｋ≧３）の検証パターンの中から１つの検証パターンを選択する検証パターン選択ステップと、
前記検証パターン選択ステップで選択された検証パターンの情報を前記証明装置に対して送信する検証パターン送信ステップと、
ｋ通りの回答情報の中から、前記検証パターン送信ステップで送信された検証パターンの情報に対応する回答情報を前記証明装置から受信する回答受信ステップと、
前記メッセージ受信ステップで受信されたメッセージｃ、及び前記回答受信ステップで受信された回答情報を用いて前記証明装置の正当性を検証する検証ステップと、
を含み、
前記回答情報は、前記ｋ通りの回答情報を用いて実施した前記メッセージｃに対するｋ通りの検証パターンが全て成功した場合に秘密鍵ｓが計算可能となる情報である、
認証方法。
【請求項１４】
ｓ∈Ｋ^{n}を秘密鍵に設定し、環Ｋ上の多次多項式ｆ$_{i}$（ｘ$_{1}$，…，ｘ$_{n}$）（ｉ＝１～ｍ）を公開鍵またはシステムパラメータ、及びｙ$_{i}$＝ｆ$_{i}$（ｓ）を公開鍵に設定する鍵設定ステップと、
検証装置に対してメッセージｃを送信するメッセージ送信ステップと、
前記検証装置から応答αを受信する応答受信ステップと、
前記応答受信ステップで受信された応答αを用いて、前記メッセージｃに対する検証に利用される多項式ｆ”を生成する多項式生成ステップと、
前記多項式生成ステップで生成された多項式ｆ”を前記検証装置に送信する多項式送信ステップと、
１つの前記メッセージｃに対するｋ通り（ｋ≧２）の検証パターンの中から前記検証装置により選択された１つの検証パターンの情報を受信する検証パターン受信ステップと、
ｋ通りの回答情報の中から、前記検証パターン受信ステップで受信された検証パターンの情報に対応する回答情報を前記検証装置に送信する回答送信ステップと、
を含み、
前記回答情報は、２通りの前記応答α、前記多項式ｆ”及び前記ｋ通りの回答情報を用いて実施した前記メッセージｃに対する計２ｋ通りの応答と検証パターンの組合せが全て成功した場合に秘密鍵ｓが計算可能となる情報である、
認証方法。

【請求項１５】
ｓ∈Ｋ^{n}が秘密鍵に設定され、環Ｋ上の多次多項式ｆ$_{i}$（ｘ$_{1}$，…，ｘ$_{n}$）（ｉ＝１～ｍ）が公開鍵またはシステムパラメータ、及びｙ$_{i}$＝ｆ$_{i}$（ｓ）が公開鍵に設定されており、
証明装置からメッセージｃを受信するメッセージ受信ステップと、
前記証明装置に対して応答αを送信する応答送信ステップと、
前記応答送信ステップで送信された応答αを用いて前記証明装置により生成された、前記メッセージｃに対する検証に利用される多項式ｆ”を受信する多項式受信ステップと、
１つの前記メッセージｃに対するｋ通り（ｋ≧２）の検証パターンの中から１つの検証パターンを選択する検証パターン選択ステップと、
前記検証パターン選択ステップで選択された検証パターンの情報を前記証明装置に対して送信する検証パターン送信ステップと、
ｋ通りの回答情報の中から、前記検証パターン送信ステップで送信された検証パターンの情報に対応する回答情報を前記証明装置から受信する回答受信ステップと、
前記メッセージ受信ステップで受信されたメッセージｃ、前記多項式受信ステップで受信された多項式ｆ”、及び前記回答受信ステップで受信された回答情報を用いて前記証明装置の正当性を検証する検証ステップと、
を含み、
前記回答情報は、２通りの前記応答α、前記多項式ｆ”及び前記ｋ通りの回答情報を用いて実施した前記メッセージｃに対する計２ｋ通りの応答と検証パターンの組合せが全て成功した場合に秘密鍵ｓが計算可能となる情報である、
認証方法。
【請求項１６】
ｓ∈Ｋ^{n}を秘密鍵に設定し、環Ｋ上の多次多項式ｆ$_{i}$（ｘ$_{1}$，…，ｘ$_{n}$）（ｉ＝１～ｍ）を公開鍵またはシステムパラメータ、及びｙ$_{i}$＝ｆ$_{i}$（ｓ）を公開鍵に設定する鍵設定機能と、
検証装置に対してメッセージｃを送信するメッセージ送信機能と、
１つの前記メッセージｃに対するｋ通り（ｋ≧３）の検証パターンの中から前記検証装置により選択された１つの検証パターンの情報を受信する検証パターン受信機能と、
ｋ通りの回答情報の中から、前記検証パターン受信機能により受信された検証パターンの情報に対応する回答情報を前記検証装置に送信する回答送信機能と、
をコンピュータに実現させるためのプログラムであり、
前記回答情報は、前記ｋ通りの回答情報を用いて実施した前記メッセージｃに対するｋ通りの検証パターンが全て成功した場合に秘密鍵ｓが計算可能となる情報である、
プログラム。
【請求項１７】
ｓ∈Ｋ^{n}を秘密鍵に設定し、環Ｋ上の多次多項式ｆ$_{i}$（ｘ$_{1}$，…，ｘ$_{n}$）（ｉ＝１～ｍ）を公開鍵またはシステムパラメータ、及びｙ$_{i}$＝ｆ$_{i}$（ｓ）を公開鍵に設定する鍵設定部と、
前記多次多項式ｆ$_{i}$（ｘ$_{1}$，…，ｘ$_{n}$）及び前記秘密鍵ｓに基づいてＮ個のメッセージｃを生成するメッセージ生成部と、
文書Ｍ及び前記メッセージｃを一方向性関数に適用して得られた情報に基づいて、ｋ^{N}通り（ｋ≧３）の検証パターンの中から検証パターンを選択する検証パターン選択部と、
前記検証パターン選択部により選択された検証パターンに応じて、前記メッセージｃ及び前記文書Ｍを用いた検証を通過するような電子署名σを生成する署名生成部と、
を備え、
前記電子署名σは、前記（ｋ－１）^{N}＋１通りの検証パターンに対応する電子署名σを用いて実施した検証が全て成功した場合に秘密鍵ｓが計算可能となる情報である、
署名生成装置。
【請求項１８】

前記メッセージ生成部により生成された単数又は複数のメッセージｃを送信する第１ステップ、それぞれのメッセージｃに対して前記検証パターン選択部により検証パターンを選択する第２ステップ、それぞれの検証パターンの情報に対して前記署名生成部により電子署名σを送信する第３ステップを実行し、電子署名σを用いて実施した検証が全て成功した場合に認証が成功する、
請求項１７に記載の署名生成装置。
【請求項１９】
前記ｍ及びｎは、ｍ＜ｎの関係を有する、
請求項１７に記載の署名生成装置。
【請求項２０】
前記多次多項式f_iは２次多項式である、
請求項１７に記載の署名生成装置。
【請求項２１】
$s \in K^n$を秘密鍵に設定し、環Ｋ上の多次多項式$f_i(x_1, \cdots, x_n)$（ｉ＝１～ｍ）を公開鍵またはシステムパラメータ、及び$y_i = f_i(s)$を公開鍵に設定する鍵設定部と、
検証装置に対してメッセージｃを送信するメッセージ送信部と、
１つの前記メッセージｃに対するｋ通り（ｋ≧３）の検証パターンの中から前記検証装置により選択された１つの検証パターンの情報を受信する検証パターン受信部と、
ｋ通りの回答情報の中から、前記検証パターン受信部により受信された検証パターンの情報に対応する回答情報を前記検証装置に送信する回答送信部と、
を備え、
前記回答情報は、前記秘密鍵ｓを$r \in K^n$によりマスクした情報$z \in K^n$と、前記ｒを$t \in K^n$によりマスクした$t' \in K^n$と、前記多次多項式f_iに前記ｒを代入した$f_i(r)$を$e_i \in K$によりマスクした$e_i' \in K$を用いて計算される情報である、
認証装置。
【請求項２２】
前記メッセージ送信部により単数又は複数のメッセージｃを送信する第１ステップ、それぞれのメッセージｃに対して前記検証パターン受信部により前記検証装置からそれぞれ検証パターンの情報を受信する第２ステップ、それぞれの検証パターンの情報に対して前記回答送信部により回答情報を送信する第３ステップを実行し、前記検証装置により全ての回答情報で検証が成功した場合に認証が成功する、
請求項２１に記載の認証装置。
【請求項２３】
前記多次多項式f_iは２次多項式である、
請求項１、５、６、９、２１のいずれか１項に記載の認証装置。

【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、認証装置、認証方法、プログラム、及び署名生成装置に関する。
【背景技術】
【０００２】
情報処理技術や通信技術の急速な発展に伴い、公文書、私文書を問わず、文書の電子化が急速に進んでいる。これに伴い、多くの個人や企業は、電子文書の安全管理に大きな関心を寄せている。こうした関心の高まりを受け、各方面で電子文書の盗聴や偽造等のタンパリング行為に対する安全性が盛んに議論されるようになってきた。電子文書の盗聴に対する安全性は、例えば、電子文書を暗号化することにより確保される。また、電子文書の偽造に対する安全性は、例えば、電子署名を利用することにより確保される。但し、暗号や電子署名には十分なタンパリング耐性が求められる。

【０００３】
電子署名は、電子文書の作成者を特定するために利用される。そのため、電子署名は、電子文書の作成者しか生成できないようにすべきである。仮に、悪意ある第三者が同じ電子署名を生成できてしまうと、その第三者が電子文書の作成者に成りすますことができてしまう。つまり、悪意ある第三者により電子文書が偽造されてしまう。こうした偽造を防止するため、電子署名の安全性については様々な議論が交わされてきた。現在広く利用されている電子署名方式としては、例えば、ＲＳＡ署名方式やＤＳＡ署名方式を利用する方式が知られている。
【０００４】
ＲＳＡ署名方式は、「大きな合成数に対する素因数分解の困難性（以下、素因数分解問題）」を安全性の根拠とする。また、ＤＳＡ署名方式は、「離散対数問題に対する解の導出の困難性」を安全性の根拠とする。これらの根拠は、古典的なコンピュータを利用して素因数分解問題や離散対数問題を効率的に解くアルゴリズムが存在しないことに起因する。つまり、上記の困難性は、古典的なコンピュータにおける計算量的な困難性を意味する。しかしながら、量子コンピュータを用いると、素因数分解問題や離散対数問題に対する解答が効率的に算出されてしまうと言われている。
【０００５】
現在利用されている電子署名方式や公開鍵認証方式の多くは、ＲＳＡ署名方式やＤＳＡ署名方式と同様、素因数分解問題や離散対数問題の困難性に安全性の根拠をおいている。そのため、こうした電子署名方式や公開鍵認証方式は、量子コンピュータが実用化された場合に、その安全性が確保されないことになる。そこで、素因数分解問題や離散対数問題など、量子コンピュータにより容易に解かれてしまう問題とは異なる問題に安全性の根拠をおく新たな電子署名方式及び公開鍵認証方式が求められている。量子コンピュータにより容易に解くことが難しい問題としては、例えば、多変数多項式に対する解答の困難性（以下、多変数多項式問題）がある。
【０００６】
多変数多項式問題に安全性の根拠をおく電子署名方式としては、例えば、ＭＩ（Ｍａｔｓｕｍｏｔｏ－Ｉｍａｉ ｃｒｙｐｔｏｇｒａｐｈｙ）、ＨＦＥ（Ｈｉｄｄｅｎ Ｆｉｅｌｄ Ｅｑｕａｔｉｏｎ ｃｒｙｐｔｏｇｒａｐｈｙ）、ＯＶ（Ｏｉｌ－Ｖｉｎｅｇａｒ ｓｉｇｎａｔｕｒｅ ｓｃｈｅｍｅ）、ＴＴＭ（Ｔａｍｅｄ Ｔｒａｎｓｆｏｒｍａｔｉｏｎ Ｍｅｔｈｏｄ ｃｒｙｐｔｏｇｒａｐｈｙ）に基づくものが知られている。例えば、下記の非特許文献１、２には、ＨＦＥに基づく電子署名方式が開示されている。
【先行技術文献】
【非特許文献】
【０００７】
【非特許文献１】Ｊａｃｑｕｅｓ Ｐａｔａｒｉｎ Ａｓｙｍｍｅｔｒｉｃ Ｃｒｙｐｔｏｇｒａｐｈｙ ｗｉｔｈ ａ Ｈｉｄｄｅｎ Ｍｏｎｏｍｉａｌ. ＣＲＹＰＴＯ １９９６, ｐｐ. ４５－６０.
【非特許文献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.
【発明の概要】
【発明が解決しようとする課題】
【０００８】
上記の通り、多変数多項式問題は、量子コンピュータを用いても解くことが困難なＮＰ困難問題と呼ばれる問題の一例である。通常、ＨＦＥなどに代表される多変数多項式問題

を利用した公開鍵認証方式は、特殊なトラップドアが仕込まれた多次多変数連立方程式を利用している。例えば、$x_1, \cdots, x_n$に関する多次多変数連立方程式$F(x_1, \cdots, x_n) = y$と線形変換A，Bが用意され、線形変換A，Bが秘密に管理される。この場合、多次多変数連立方程式F、線形変換A，Bがトラップドアとなる。

【０００９】

トラップドアF，A，Bを知っているエンティティは、$x_1, \cdots, x_n$に関する方程式$B(F(A(x_1, \cdots, x_n))) = y'$を解くことができる。一方、トラップドアF，A，Bを知らないエンティティは、$x_1, \cdots, x_n$に関する方程式$B(F(A(x_1, \cdots, x_n))) = y'$を解くことができない。この仕組みを利用することにより、多次多変数連立方程式の解答困難性を安全性の根拠とする公開鍵認証方式や電子署名方式を実現することができる。

【００１０】

上記の通り、こうした公開鍵認証方式や電子署名方式を実現するには、$B(F(A(x_1, \cdots, x_n))) = y$を満たすような特殊な多次多変数連立方程式を利用する必要がある。しかし、従来方式では署名生成時に多次多変数連立方程式Fを解く必要があるため，Fには比較的容易に解けるものしか用いることができない。すなわち、こうした従来方式では比較的容易に解ける３つの関数（トラップドア）B、F、Aを合成した型式の多次多変数連立方程式$B(F(A(x_1, \cdots, x_n))) = y$しか用いることができないため、十分な安全性を確保することが難しい。その結果、このような効率的に解く手段（トラップドア）を持つ多次多変数連立方程式を用いた公開鍵認証方式や電子署名方式に対して数々の攻撃法が提案されており、効率的に解く手段（トラップドア）を持たない多次多変数連立方程式を公開鍵認証方式や電子署名方式に用いることができるようにする工夫が求められている。

【００１１】

そこで、本発明は、上記問題に鑑みてなされたものであり、本発明の目的とするところは、効率的に解く手段（トラップドア）を持たない多次多変数連立方程式を用いて安全性を高めた公開鍵認証方式又は電子署名方式を実現することが可能な、新規かつ改良された認証装置、認証方法、プログラム、及び署名生成装置を提供することにある。

【００１２】

上記課題を解決するために、本発明のある観点によれば、$s \in K^n$を秘密鍵に設定し、環K上の多次多項式$f_i(x_1, \cdots, x_n)$（$i = 1 \sim m$）及び$y_i = f_i(s)$を公開鍵に設定する鍵設定部と、検証者に対してメッセージcを送信するメッセージ送信部と、１つの前記メッセージcに対するk通り（$k \geqq 3$）の検証パターンの中から前記検証者により選択された１つの検証パターンの情報を受信する検証パターン受信部と、k通りの回答情報の中から、前記検証パターン受信部により受信された検証パターンの情報に対応する回答情報を前記検証者に送信する回答送信部と、を備え、前記回答情報は、前記k通りの回答情報を用いて実施した前記メッセージcに対するk通りの検証パターンが全て成功した場合に秘密鍵sが計算可能となる情報である、認証装置が提供される。

【００１３】

また、上記の認証装置は、前記メッセージ送信部により複数のメッセージcをk回送信する第１ステップ、それぞれのメッセージcに対して前記検証パターン受信部により前記検証者からそれぞれ検証パターンの情報を受信する第２ステップ、それぞれの検証パターンの情報に対して前記回答送信部により回答情報を送信する第３ステップを実行し、前記検証者により全ての回答情報で検証が成功した場合に認証が成功する、ように構成されていてもよい。

【００１４】

また、上記の認証装置は、前記メッセージ送信部により複数のメッセージcを送信する第１ステップ、それぞれのメッセージcに対して前記検証パターン受信部により前記検証者からそれぞれ検証パターンの情報を受信する第２ステップ、それぞれの検証パターンの情報に対して前記回答送信部により回答情報を送信する第３ステップを実行する処理を繰

り返し、前記第１～第３ステップを所定回数実行した結果、前記検証者により毎回全ての回答情報で検証が成功した場合に認証が成功する、ように構成されていてもよい。

【００１５】

また、前記メッセージｃがｃ＝（c_1，…，c_m）である場合、前記メッセージ送信部は、一方向性関数Ｈを用いて新たなメッセージｃ’＝Ｈ（ｃ）を算出して、前記検証者に対してメッセージｃ’を送信し、前記回答送信部は、前記回答情報と共に、当該回答情報を利用しても前記検証者が復元することができないメッセージｃの要素を送信する、ように構成されていてもよい。

【００１６】

また、上記課題を解決するために、本発明の別の観点によれば、$s \in K^n$が秘密鍵に設定され、環Ｋ上の多次多項式f_i（x_1，…，x_n）（ｉ＝１～ｍ）及び$y_i = f_i$（ｓ）が公開鍵に設定されており、証明者からメッセージｃを受信するメッセージ受信部と、１つの前記メッセージｃに対するｋ通り（ｋ≧３）の検証パターンの中から１つの検証パターンを選択する検証パターン選択部と、前記検証パターン選択部により選択された検証パターンの情報を前記証明者に対して送信する検証パターン送信部と、ｋ通りの回答情報の中から、前記検証パターン送信部により送信された検証パターンの情報に対応する回答情報を前記証明者から受信する回答受信部と、前記メッセージ受信部により受信されたメッセージｃ、及び前記回答受信部により受信された回答情報を用いて前記証明者の正当性を検証する検証部と、を備え、前記回答情報は、前記ｋ通りの回答情報を用いて実施した前記メッセージｃに対するｋ通りの検証パターンが全て成功した場合に秘密鍵ｓが計算可能となる情報である、認証装置が提供される。

【００１７】

また、上記課題を解決するために、本発明の別の観点によれば、$s \in K^n$を秘密鍵に設定し、環Ｋ上の多次多項式f_i（x_1，…，x_n）（ｉ＝１～ｍ）及び$y_i = f_i$（ｓ）を公開鍵に設定する鍵設定部と、検証者に対してメッセージｃを送信するメッセージ送信部と、前記検証者から応答αを受信する応答受信部と、前記応答受信部により受信された応答αを用いて、前記メッセージｃに対する検証に利用される多項式の組Ｆ”を生成する多項式生成部と、前記多項式生成部により生成された多項式の組Ｆ”を前記検証者に送信する多項式送信部と、１つの前記メッセージｃに対するｋ通り（ｋ≧２）の検証パターンの中から前記検証者により選択された１つの検証パターンの情報を受信する検証パターン受信部と、ｋ通りの回答情報の中から、前記検証パターン受信部により受信された検証パターンの情報に対応する回答情報を前記検証者に送信する回答送信部と、を備え、前記回答情報は、２通りの前記応答α、前記多項式ｆ”及び前記ｋ通りの回答情報を用いて実施した前記メッセージｃに対する計２ｋ通りの応答と検証パターンの組合せが全て成功した場合に秘密鍵ｓが計算可能となる情報である、認証装置が提供される。

【００１８】

また、上記の認証装置は、前記メッセージ送信部により複数のメッセージｃを送信する第１ステップ、それぞれのメッセージｃに対して前記応答受信部から応答αを受信する第２ステップ、当該第２ステップで受信した応答αを用いて前記多項式生成部によりそれぞれ多項式ｆ”を生成し、前記多項式送信部により当該多項式ｆ”を送信する第３ステップ、それぞれのメッセージｃに対して前記検証パターン受信部により前記検証者からそれぞれ検証パターンの情報を受信する第４ステップ、それぞれの検証パターンの情報に対して前記回答送信部により回答情報を送信する第５ステップを実行し、前記検証者により全ての回答情報で検証が成功した場合に認証が成功する、ように構成されていてもよい。

【００１９】

また、上記の認証装置は、前記メッセージ送信部により複数のメッセージｃを送信する第１ステップ、それぞれのメッセージｃに対して前記応答受信部から応答αを受信する第２ステップ、当該第２ステップで受信した応答αを用いて前記多項式生成部によりそれぞれ多項式ｆ”を生成し、前記多項式送信部により当該多項式ｆ”を送信する第３ステップ、それぞれのメッセージｃに対して前記検証パターン受信部により前記検証者からそれぞれ

検証パターンの情報を受信する第４ステップ、それぞれの検証パターンの情報に対して前記回答送信部により回答情報を送信する第５ステップを実行する処理を繰り返し、前記第１～第５ステップを所定回数実行した結果、前記検証者により毎回全ての回答情報で検証が成功した場合に認証が成功する、ように構成されていてもよい。

【００２０】

また、上記課題を解決するために、本発明の別の観点によれば、$s \in K^n$が秘密鍵に設定され、環Ｋ上の多次多項式$f_i(x_1, \cdots, x_n)$（$i = 1 \sim m$）及び$y_i = f_i(s)$が公開鍵に設定されており、証明者からメッセージｃを受信するメッセージ受信部と、前記証明者に対して応答αを送信する応答送信部と、前記応答送信部により送信された応答αを用いて前記証明者により生成された、前記メッセージｃに対する検証に利用される多項式ｆ”を受信する多項式受信部と、１つの前記メッセージｃに対するｋ通り（$k \geqq 2$）の検証パターンの中から１つの検証パターンを選択する検証パターン選択部と、前記検証パターン選択部により選択された検証パターンの情報を前記証明者に対して送信する検証パターン送信部と、ｋ通りの回答情報の中から、前記検証パターン送信部により送信された検証パターンの情報に対応する回答情報を前記証明者から受信する回答受信部と、前記メッセージ受信部により受信されたメッセージｃ、前記多項式受信部により受信された多項式ｆ”、及び前記回答受信部により受信された回答情報を用いて前記証明者の正当性を検証する検証部と、を備え、前記回答情報は、２通りの前記応答α、前記多項式ｆ”及び前記ｋ通りの回答情報を用いて実施した前記メッセージｃに対する計２ｋ通りの応答と検証パターンの組合せが全て成功した場合に秘密鍵ｓが計算可能となる情報である、認証装置が提供される。

【００２１】

また、上記課題を解決するために、本発明の別の観点によれば、$s \in K^n$を秘密鍵に設定し、環Ｋ上の多次多項式$f_i(x_1, \cdots, x_n)$（$i = 1 \sim m$）及び$y_i = f_i(s)$を公開鍵に設定する鍵設定ステップと、検証者に対してメッセージｃを送信するメッセージ送信ステップと、１つの前記メッセージｃに対するｋ通り（$k \geqq 3$）の検証パターンの中から前記検証者により選択された１つの検証パターンの情報を受信する検証パターン受信ステップと、ｋ通りの回答情報の中から、前記検証パターン受信ステップで受信された検証パターンの情報に対応する回答情報を前記検証者に送信する回答送信ステップと、を含み、前記回答情報は、前記ｋ通りの回答情報を用いて実施した前記メッセージｃに対するｋ通りの検証パターンが全て成功した場合に秘密鍵ｓが計算可能となる情報である、認証方法が提供される。

【００２２】

また、上記課題を解決するために、本発明の別の観点によれば、$s \in K^n$が秘密鍵に設定され、環Ｋ上の多次多項式$f_i(x_1, \cdots, x_n)$（$i = 1 \sim m$）及び$y_i = f_i(s)$が公開鍵に設定されており、証明者からメッセージｃを受信するメッセージ受信ステップと、１つの前記メッセージｃに対するｋ通り（$k \geqq 3$）の検証パターンの中から１つの検証パターンを選択する検証パターン選択ステップと、前記検証パターン選択ステップで選択された検証パターンの情報を前記証明者に対して送信する検証パターン送信ステップと、ｋ通りの回答情報の中から、前記検証パターン送信ステップで送信された検証パターンの情報に対応する回答情報を前記証明者から受信する回答受信ステップと、前記メッセージ受信ステップで受信されたメッセージｃ、及び前記回答受信ステップで受信された回答情報を用いて前記証明者の正当性を検証する検証ステップと、を含み、前記回答情報は、前記ｋ通りの回答情報を用いて実施した前記メッセージｃに対するｋ通りの検証パターンが全て成功した場合に秘密鍵ｓが計算可能となる情報である、認証方法が提供される。

【００２３】

また、上記課題を解決するために、本発明の別の観点によれば、$s \in K^n$を秘密鍵に設定し、環Ｋ上の多次多項式$f_i(x_1, \cdots, x_n)$（$i = 1 \sim m$）及び$y_i = f_i(s)$を公開鍵に設定する鍵設定ステップと、検証者に対してメッセージｃを送信するメッセージ送信ステップと、前記検証者から応答αを受信する応答受信ステップと、前記応答受信

ステップで受信された応答αを用いて、前記メッセージｃに対する検証に利用される多項式ｆ”を生成する多項式生成ステップと、前記多項式生成ステップで生成された多項式ｆ”を前記検証者に送信する多項式送信ステップと、１つの前記メッセージｃに対するｋ通り（ｋ≧２）の検証パターンの中から前記検証者により選択された１つの検証パターンの情報を受信する検証パターン受信ステップと、ｋ通りの回答情報の中から、前記検証パターン受信ステップで受信された検証パターンの情報に対応する回答情報を前記検証者に送信する回答送信ステップと、を含み、前記回答情報は、２通りの前記応答α、前記多項式ｆ”及び前記ｋ通りの回答情報を用いて実施した前記メッセージｃに対する計２ｋ通りの応答と検証パターンの組合せが全て成功した場合に秘密鍵ｓが計算可能となる情報である、認証装置が提供される。

【００２４】

また、上記課題を解決するために、本発明の別の観点によれば、$s \in K^n$が秘密鍵に設定され、環Ｋ上の多次多項式$f_i(x_1, \cdots, x_n)$（ｉ＝１～ｍ）及び$y_i = f_i(s)$が公開鍵に設定されており、証明者からメッセージｃを受信するメッセージ受信ステップと、前記証明者に対して応答αを送信する応答送信ステップと、前記応答送信ステップで送信された応答αを用いて前記証明者により生成された、前記メッセージｃに対する検証に利用される多項式ｆ”を受信する多項式受信ステップと、１つの前記メッセージｃに対するｋ通り（ｋ≧２）の検証パターンの中から１つの検証パターンを選択する検証パターン選択ステップと、前記検証パターン選択ステップで選択された検証パターンの情報を前記証明者に対して送信する検証パターン送信ステップと、ｋ通りの回答情報の中から、前記検証パターン送信ステップで送信された検証パターンの情報に対応する回答情報を前記証明者から受信する回答受信ステップと、前記メッセージ受信ステップで受信されたメッセージｃ、前記多項式受信ステップで受信された多項式ｆ”、及び前記回答受信ステップで受信された回答情報を用いて前記証明者の正当性を検証する検証ステップと、を含み、前記回答情報は、２通りの前記応答α、前記多項式ｆ”及び前記ｋ通りの回答情報を用いて実施した前記メッセージｃに対する計２ｋ通りの応答と検証パターンの組合せが全て成功した場合に秘密鍵ｓが計算可能となる情報である、認証装置が提供される。

【００２５】

また、上記課題を解決するために、本発明の別の観点によれば、$s \in K^n$を秘密鍵に設定し、環Ｋ上の多次多項式$f_i(x_1, \cdots, x_n)$（ｉ＝１～ｍ）及び$y_i = f_i(s)$を公開鍵に設定する鍵設定機能と、検証者に対してメッセージｃを送信するメッセージ送信機能と、１つの前記メッセージｃに対するｋ通り（ｋ≧３）の検証パターンの中から前記検証者により選択された１つの検証パターンの情報を受信する検証パターン受信機能と、ｋ通りの回答情報の中から、前記検証パターン受信機能により受信された検証パターンの情報に対応する回答情報を前記検証者に送信する回答送信機能と、をコンピュータに実現させるためのプログラムであり、前記回答情報は、前記ｋ通りの回答情報を用いて実施した前記メッセージｃに対するｋ通りの検証パターンが全て成功した場合に秘密鍵ｓが計算可能となる情報である、プログラムが提供される。

【００２６】

さらに、上記課題を解決するために、本発明の別の観点によれば、上記のプログラムが記録された、コンピュータにより読み取り可能な記録媒体が提供される。

【００２７】

また、上記課題を解決するために、本発明の別の観点によれば、$s \in K^n$を秘密鍵に設定し、環Ｋ上の多次多項式$f_i(x_1, \cdots, x_n)$（ｉ＝１～ｍ）及び$y_i = f_i(s)$を公開鍵に設定する鍵設定部と、前記多次多項式$f_i(x_1, \cdots, x_n)$及び前記秘密鍵ｓに基づいてＮ個のメッセージｃを生成するメッセージ生成部と、文書Ｍ及び前記メッセージｃを一方向性関数に適用して得られた情報に基づいて、k^N通り（ｋ≧３）の検証パターンの中から検証パターンを選択する検証パターン選択部と、前記検証パターン選択部により選択された検証パターンに応じて、前記メッセージｃ及び前記文書Ｍを用いた検証を通過するような電子署名σを生成する署名生成部と、を備え、前記電子署名σは、前記

（ｋ－１）^{N}＋１通りの検証パターンに対応する電子署名σを用いて実施した検証が全て成功した場合に秘密鍵ｓが計算可能となる情報である、署名生成装置が提供される。

【００２８】

上記のｍ、ｎは、ｍ＜ｎであってもよい。さらに、上記のｍ、ｎは、$2^{m-n} \ll 1$であってもよい。

【発明の効果】

【００２９】

以上説明したように本発明によれば、効率的に解く手段（トラップドア）を持たない多次多変数連立方程式を用いて安全性を高めた公開鍵認証方式又は電子署名方式を実現することが可能になる。

【図面の簡単な説明】

【００３０】

【図１】公開鍵認証方式のアルゴリズム構成について説明するための説明図である。

【図２】電子署名方式のアルゴリズム構成について説明するための説明図である。

【図３】ｎパスの公開鍵認証方式について説明するための説明図である。

【図４】本発明の第１実施形態（３パス）に係る公開鍵認証方式のアルゴリズムについて説明するための説明図である。

【図５】同実施形態に係る公開鍵認証方式の拡張アルゴリズムについて説明するための説明図である。

【図６】同実施形態に係る公開鍵認証方式の並列化アルゴリズムについて説明するための説明図である。

【図７】同実施形態に係る公開鍵認証方式の非対話型アルゴリズムについて説明するための説明図である。

【図８】同実施形態に係る公開鍵認証方式の具体的なアルゴリズムについて説明するための説明図である。

【図９】同実施形態に係る公開鍵認証方式の効率的なアルゴリズムについて説明するための説明図である。

【図１０】同実施形態に係る公開鍵認証方式の効率的なアルゴリズム（変形例Ａ）について説明するための説明図である。

【図１１】同実施形態に係る公開鍵認証方式の効率的なアルゴリズム（変形例Ｂ）について説明するための説明図である。

【図１２】本発明の第２実施形態（５パス）に係る公開鍵認証方式のアルゴリズムについて説明するための説明図である。

【図１３】同実施形態に係る公開鍵認証方式の拡張アルゴリズムについて説明するための説明図である。

【図１４】同実施形態に係る公開鍵認証方式の並列化アルゴリズムについて説明するための説明図である。

【図１５】同実施形態に係る公開鍵認証方式の拡張アルゴリズムに対する並列化について説明するための説明図である。

【図１６】同実施形態に係る公開鍵認証方式の非対話型アルゴリズムについて説明するための説明図である。

【図１７】同実施形態に係る公開鍵認証方式の具体的なアルゴリズムについて説明するための説明図である。

【図１８】同実施形態に係る公開鍵認証方式の効率的なアルゴリズムについて説明するための説明図である。

【図１９】同実施形態に係る公開鍵認証方式の効率的なアルゴリズム（変形例Ａ）について説明するための説明図である。

【図２０】同実施形態に係る公開鍵認証方式の効率的なアルゴリズム（変形例Ｂ）について説明するための説明図である。

【図２１】同実施形態に係る公開鍵認証方式の効率的なアルゴリズム（変形例Ｃ）につい

て説明するための説明図である。
【図２２】同実施形態に係る公開鍵認証方式の効率的なアルゴリズム（変形例Ｄ）について説明するための説明図である。
【図２３】同実施形態に係る公開鍵認証方式の効率的なアルゴリズム（変形例Ｅ）について説明するための説明図である。
【図２４】同実施形態に係る公開鍵認証方式の効率的なアルゴリズム（変形例Ｆ）について説明するための説明図である。
【図２５】同実施形態に係る公開鍵認証方式の効率的なアルゴリズム（変形例Ｇ）について説明するための説明図である。
【図２６】本発明の各実施形態に係るアルゴリズムを実行することが可能な情報処理装置のハードウェア構成例について説明するための説明図である。
【図２７】本発明の第１及び第２実施形態に係る公開鍵認証方式の効率を比較した図表である。
【図２８】本発明の第１及び第２実施形態に係る公開鍵認証方式において用いられるパラメータの好適な設定方法及びその効果について説明するための説明図である。
【発明を実施するための形態】
【００３１】
以下に添付図面を参照しながら、本発明の好適な実施の形態について詳細に説明する。なお、本明細書及び図面において、実質的に同一の機能構成を有する構成要素については、同一の符号を付することにより重複説明を省略する。
【００３２】
［説明の流れについて］
ここで、以下に記載する本発明の実施形態に関する説明の流れについて簡単に述べる。まず、図１を参照しながら、公開鍵認証方式のアルゴリズム構成について説明する。次いで、図２を参照しながら、電子署名方式のアルゴリズム構成について説明する。次いで、図３を参照しながら、ｎパスの公開鍵認証方式について説明する。なお、本実施形態に係る説明に先立ち、一般的なＨＦＥ署名方式についても簡単に説明する。
【００３３】
次いで、図４を参照しながら、本発明の第１実施形態（３パス）に係る公開鍵認証方式のアルゴリズムについて説明する。次いで、図５を参照しながら、同実施形態に係る公開鍵認証方式の拡張アルゴリズムについて説明する。次いで、図６を参照しながら、同実施形態に係る公開鍵認証方式の並列化アルゴリズムについて説明する。次いで、図７を参照しながら、同実施形態に係る公開鍵認証方式の非対話型アルゴリズムについて説明する。次いで、図８を参照しながら、同実施形態に係る公開鍵認証方式の具体的なアルゴリズムについて説明する。次いで、図９～図１１を参照しながら、同実施形態に係る公開鍵認証方式の効率的なアルゴリズムについて説明する。
【００３４】
次いで、図１２を参照しながら、本発明の第２実施形態（５パス）に係る公開鍵認証方式のアルゴリズムについて説明する。次いで、図１３を参照しながら、同実施形態に係る公開鍵認証方式の拡張アルゴリズムについて説明する。次いで、図１４、図１５を参照しながら、同実施形態に係る公開鍵認証方式の並列化アルゴリズムについて説明する。次いで、図１６を参照しながら、同実施形態に係る公開鍵認証方式の非対話型アルゴリズムについて説明する。次いで、図１７を参照しながら、同実施形態に係る公開鍵認証方式の具体的なアルゴリズムについて説明する。次いで、図１８～図２５を参照しながら、同実施形態に係る公開鍵認証方式の効率的なアルゴリズムについて説明する。
【００３５】
次いで、本発明の第１及び第２実施形態に係る効率的なアルゴリズムを２次以上の多変数多項式に適用するための拡張手法について説明する。次いで、図２６を参照しながら、本発明の第１及び第２実施形態に係る各アルゴリズムを実現することが可能な情報処理装置のハードウェア構成例について説明する。最後に、本実施形態の技術的思想について纏

め、当該技術的思想から得られる作用効果について簡単に説明する。

【００３６】

（説明項目）

【００３７】

＜１：はじめに＞

はじめに、本発明に係る実施形態について詳細に説明するに先立ち、一般的な公開鍵認証方式のアルゴリズム構成、一般的な電子署名方式のアルゴリズム構成、及びｎパスの公開鍵認証方式について簡単に説明する。

【００３８】

［１－１：公開鍵認証方式のアルゴリズム構成］

まず、図１を参照しながら、一般的な公開鍵認証方式のアルゴリズム構成について説明する。図１は、一般的な公開鍵認証方式のアルゴリズム構成について説明するための説明図である。

【００３９】

（概要）

公開鍵認証方式とは、ある人（証明者）が、公開鍵ｐｋ及び秘密鍵ｓｋを利用して、他の人（検証者）に本人であることを納得させるための認証方式である。例えば、証明者Ａの公開鍵pk_Aは、検証者に公開される。一方、証明者Ａの秘密鍵sk_Aは、証明者により秘密に管理される。公開鍵認証方式では、公開鍵pk_Aに対応する秘密鍵sk_Aを知る者が証明者Ａ本人であるとみなされる。

【００４０】
証明者Ａが検証者Ｂに対して本人であることを証明しようとする場合、証明者Ａは、検証者Ｂと対話プロトコルを実行して、自身が公開鍵pk_Aに対応する秘密鍵sk_Aを知っていることを証明すればよい。そして、対話プロトコルにより証明者Ａが秘密鍵sk_Aを知っていることが検証者Ｂにより証明された場合、証明者Ａの正当性（本人であること）が証明される。
【００４１】
なお、公開鍵認証方式の安全性を確保するには、次に示す２つの条件が求められる。
【００４２】
１つ目の条件は、対話プロトコルを実行した際に秘密鍵ｓｋを持たない偽証者により偽証が成立してしまう確率を限りなく小さくすることである。この１つ目の条件が成り立つことを「健全性」と呼ぶ。つまり、健全性を有する対話プロトコルにおいては、秘密鍵ｓｋを持たない偽証者により、無視できない確率で偽証が成立することはないと言い換えられる。２つ目の条件は、対話プロトコルを実行したとしても、証明者Ａが有する秘密鍵sk_Aの情報が検証者Ｂに一切漏れることがないようにすることである。この２つ目の条件が成り立つことを「零知識性」と呼ぶ。
【００４３】
上記の健全性と零知識性を有する対話プロトコルを利用することにより、公開鍵認証方式の安全性が確保される。
【００４４】
（モデル）
公開鍵認証方式のモデルには、図１に示すように、証明者と検証者という２つのエンティティが存在する。証明者は、鍵生成アルゴリズムＧｅｎを用いて、証明者固有の秘密鍵ｓｋと公開鍵ｐｋの組を生成する。次いで、証明者は、鍵生成アルゴリズムＧｅｎを用いて生成した秘密鍵ｓｋと公開鍵ｐｋの組を利用して検証者と対話プロトコルを実行する。このとき、証明者は、証明者アルゴリズムＰを利用して対話プロトコルを実行する。上記の通り、対話プロトコルにおいて、証明者は、証明者アルゴリズムＰを利用して、秘密鍵ｓｋを保有していることを検証者に証明する。
【００４５】
一方、検証者は、検証者アルゴリズムＶを利用して対話プロトコルを実行し、証明者が公開している公開鍵に対応する秘密鍵を、その証明者が保有しているか否かを検証する。つまり、検証者は、証明者が公開鍵に対応する秘密鍵を保有しているか否かを検証するエンティティである。このように、公開鍵認証方式のモデルは、証明者と検証者という２つのエンティティ、及び、鍵生成アルゴリズムＧｅｎ、証明者アルゴリズムＰ、検証者アルゴリズムＶという３つのアルゴリズムにより構成される。
【００４６】
なお、以下の説明において、「証明者」「検証者」という表現を用いるが、これらの表現はあくまでもエンティティを意味するものである。従って、鍵生成アルゴリズムＧｅｎ、証明者アルゴリズムＰを実行する主体は、「証明者」のエンティティに対応する情報処理装置である。同様に、検証者アルゴリズムＶを実行する主体は、情報処理装置である。これら情報処理装置のハードウェア構成は、例えば、図２６に示した通りである。つまり、鍵生成アルゴリズムＧｅｎ、証明者アルゴリズムＰ、検証者アルゴリズムＶは、ＲＯＭ９０４、ＲＡＭ９０６、記憶部９２０、リムーバブル記録媒体９２８などに記録されたプログラムに基づいてＣＰＵ９０２により実行される。
【００４７】
（鍵生成アルゴリズムＧｅｎ）
鍵生成アルゴリズムＧｅｎは、証明者により利用される。そして、鍵生成アルゴリズムＧｅｎは、証明者に固有の秘密鍵ｓｋと公開鍵ｐｋの組を生成するアルゴリズムである。鍵生成アルゴリズムＧｅｎにより生成された公開鍵ｐｋは公開される。そして、公開された公開鍵ｐｋは、検証者により利用される。一方、鍵生成アルゴリズムＧｅｎにより生成

された秘密鍵ｓｋは、証明者が秘密に管理する。そして、秘密に管理される秘密鍵ｓｋは、検証者に対して公開鍵ｐｋに対応する秘密鍵ｓｋを保有していることを証明するために利用される。形式的に、鍵生成アルゴリズムＧｅｎは、セキュリティパラメータ1^{λ}（λは０以上の整数）を入力とし、秘密鍵ｓｋと公開鍵ｐｋを出力するアルゴリズムとして、下記の式（１）のように表現される。

【００４８】

【数１】

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

【００４９】

（証明者アルゴリズムＰ）

証明者アルゴリズムＰは、証明者により利用される。そして、証明者アルゴリズムＰは、公開鍵ｐｋに対応する秘密鍵ｓｋを保有していることを証明するアルゴリズムである。証明者アルゴリズムＰは、証明者の秘密鍵ｓｋと公開鍵ｐｋを入力とし、検証者との対話プロトコルを実行するアルゴリズムとして定義される。

【００５０】

（検証者アルゴリズムＶ）

検証者アルゴリズムＶは、検証者により利用される。そして、検証者アルゴリズムＶは、対話プロトコルの中で、公開鍵ｐｋに対応する秘密鍵ｓｋを証明者が保有しているか否かを検証するアルゴリズムである。検証者アルゴリズムＶは、証明者の公開鍵ｐｋを入力とし、証明者との間で対話プロトコルを実行した後、０又は１（１ｂｉｔ）を出力するアルゴリズムとして定義される。なお、出力０の場合には証明者が不正なものであり、出力１の場合には証明者が正当なものであるとする。形式的に、検証者アルゴリズムＶは、下記の式（２）のように表現される。

【００５１】

【数２】

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

【００５２】

（補足）

上記の通り、公開鍵認証方式は、安全性を確保するため、健全性と零知識性という２つの条件を満たすことが求められる。しかし、証明者が秘密鍵ｓｋを保有していることを証明者に証明させるためには、証明者が秘密鍵ｓｋに依存した手続きを実行し、その結果を検証者に通知して、その通知内容に基づく検証を検証者に実行させる必要がある。秘密鍵ｓｋに依存した手続きを実行するのは、健全性を担保するために必要である。一方で、この手続きの結果を検証者に通知しても、秘密鍵ｓｋの情報が一切検証者に漏れないようにする必要がある。そのため、これらの要件を満たすように、上記の鍵生成アルゴリズムＧｅｎ、証明者アルゴリズムＰ、検証者アルゴリズムＶが設計されることが必要になる。

【００５３】

以上、一般的な公開鍵認証方式のアルゴリズム構成について説明した。

【００５４】

［１－２：電子署名方式のアルゴリズム構成］

次に、図２を参照しながら、一般的な電子署名方式のアルゴリズム構成について説明する。図２は、一般的な電子署名方式のアルゴリズム構成について説明するための説明図で

ある。

【００５５】

（概要）

紙文書とは異なり、ある電子化されたデータに対して押印したり署名を記載したりすることはできない。そのため、電子化されたデータの作成者を証明するためには、紙文書に押印したり署名を記載したりするのと同等の効果を奏する電子的な仕組みが必要とされる。このような仕組みが電子署名である。例えば、データの作成者しか知らない署名データをデータに関連付けて受領者に提供し、その署名データを受領者側で検証する仕組みのことを電子署名方式と呼ぶ。

【００５６】

（モデル）

電子署名方式のモデルには、図２に示すように、署名者と検証者という２つのエンティティが存在する。そして、電子署名方式のモデルは、鍵生成アルゴリズムＧｅｎ、署名生成アルゴリズムＳｉｇ、署名検証アルゴリズムＶｅｒという３つのアルゴリズムにより構成される。

【００５７】

署名者は、鍵生成アルゴリズムＧｅｎを利用して署名者固有の署名鍵ｓｋと検証鍵ｐｋとの組を生成する。また、署名者は、署名生成アルゴリズムＳｉｇを利用して文書Ｍに対して付与する電子署名σを生成する。つまり、署名者は、文書Ｍに電子署名を付与するエンティティである。一方、検証者は、署名検証アルゴリズムＶｅｒを利用して文書Ｍに付与された電子署名σを検証する。つまり、検証者は、文書Ｍの作成者が署名者であるか否かを確認するために、電子署名σを検証するエンティティである。

【００５８】

なお、以下の説明において、「署名者」「検証者」という表現を用いるが、これらの表現はあくまでもエンティティを意味するものである。従って、鍵生成アルゴリズムＧｅｎ、署名生成アルゴリズムＳｉｇを実行する主体は、「署名者」のエンティティに対応する情報処理装置である。同様に、署名検証アルゴリズムＶｅｒを実行する主体は、情報処理装置である。これら情報処理装置のハードウェア構成は、例えば、図２６に示した通りである。つまり、鍵生成アルゴリズムＧｅｎ、署名生成アルゴリズムＳｉｇ、署名検証アルゴリズムＶｅｒは、ＲＯＭ９０４、ＲＡＭ９０６、記憶部９２０、リムーバブル記録媒体９２８などに記録されたプログラムに基づいてＣＰＵ９０２により実行される。

【００５９】

（鍵生成アルゴリズムＧｅｎ）

鍵生成アルゴリズムＧｅｎは、署名者により利用される。そして、鍵生成アルゴリズムＧｅｎは、署名者固有の署名鍵ｓｋと検証鍵ｐｋの組を生成するアルゴリズムである。鍵生成アルゴリズムＧｅｎにより生成された検証鍵ｐｋは公開される。一方、鍵生成アルゴリズムＧｅｎにより生成された署名鍵ｓｋは、署名者により秘密に管理される。そして、署名者により秘密に管理されている署名鍵ｓｋは、文書Ｍに付与される電子署名σの生成に利用される。形式的に、鍵生成アルゴリズムＧｅｎは、セキュリティパラメータ1^{λ}（λは０以上の整数）を入力とし、秘密鍵ｓｋと公開鍵ｐｋを出力するアルゴリズムとして、下記の式（３）のように表現される。

【００６０】

【数３】

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

…（３）

【００６１】

（署名生成アルゴリズムＳｉｇ）

署名生成アルゴリズムＳｉｇは、署名者により利用される。そして、署名生成アルゴリ

ズムＳｉｇは、文書Mに対して付与される電子署名σを生成するアルゴリズムである。形式的に、署名生成アルゴリズムＳｉｇは、署名者の署名鍵ｓｋと文書Mを入力とし、下記の式（４）に示すように、電子署名σを出力するアルゴリズムとして表現される。

【００６２】

【数４】

$$\sigma \leftarrow Sig(sk, M)$$

…（４）

【００６３】

（署名検証アルゴリズムＶｅｒ）

署名検証アルゴリズムＶｅｒは、検証者により利用される。そして、署名検証アルゴリズムＶｅｒは、電子署名σが文書Mに対する正当な電子署名であるか否かを検証するアルゴリズムである。形式的に、署名検証アルゴリズムＶｅｒは、下記の式（５）に示すように、署名者の検証鍵ｐｋ、文書M、電子署名σを入力とし、０又は１（１ｂｉｔ）を出力するアルゴリズムとして表現される。なお、０を出力する場合（公開鍵ｐｋが文書Mと電子署名σを拒否する場合）、文書Mに対する電子署名σは不当である。１を出力する場合（公開鍵ｐｋが文書Mと電子署名σを受理する場合）、文書Mに対する電子署名σは正当である。

【００６４】

【数５】

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

…（５）

【００６５】

以上、一般的な電子署名方式のアルゴリズム構成について説明した。

【００６６】

［１－３：ｎパスの公開鍵認証方式］

次に、図３を参照しながら、ｎパスの公開鍵認証方式について説明する。図３は、ｎパスの公開鍵認証方式について説明するための説明図である。

【００６７】

上記の通り、公開鍵認証方式は、対話プロトコルの中で、証明者が公開鍵ｐｋに対応する秘密鍵ｓｋを保有していることを検証者に証明する認証方式である。また、公開鍵認証方式の安全性を担保するため、健全性と零知識性という２つの条件を満たす必要がある。そのため、対話プロトコルの中では、図３に示すように、証明者と検証者の双方がそれぞれ処理を実行しながら、証明者と検証者との間でｎ回の情報交換が行われる。

【００６８】

ｎパスの公開鍵認証方式の場合、証明者アルゴリズムＰを用いて証明者により処理（Ｓｔｅｐ．１）が実行され、情報T_1が検証者に送信される。次いで、検証者アルゴリズムＶを用いて検証者により処理（Ｓｔｅｐ．２）が実行され、情報T_2が証明者に送信される。同様にして処理（Ｓｔｅｐ．３，…，Ｓｔｅｐ．ｎ）が実行され、情報T_3，…，T_nが送信され、処理（Ｓｔｅｐ．ｎ＋１）が実行される。このように、情報がｎ回送受信される対話プロトコルに基づく公開鍵認証方式のことを「ｎパス」の公開鍵認証方式と呼ぶ。

【００６９】

以上、ｎパスの公開鍵認証方式について説明した。

【００７０】

［１－４：ＨＦＥ電子署名方式］

ここで、多次多変数連立方程式を用いた電子署名方式の一例として、ＨＦＥ電子署名方式について簡単に説明する。

【００７１】

（１－４－１：ＨＦＥ関数の性質）

まず、ＨＦＥ関数F_tの定義、及びＨＦＥ関数F_tの性質について簡単に説明する。

【００７２】

≪記号の定義≫

Ｋ：ｑ個の数を含む元で構成される有限環

K^n：Ｋのｎ個の直積

F_t：$K^n \to K^n$

Ａ：有限環Ｋのｎ次拡大（要素数q^n）

Ｂ：有限環Ｋのｍ次拡大（要素数q^m）

φ：線形写像$A \to K^n$（下記の式（６）を参照）

Ｓ：K^n上での可逆的アフィン変換

Ｔ：K^n上での可逆的アフィン変換

ｆ：中央写像（下記の式（７）を参照）

トラップドア：Ｓ、Ｔ、a_{ij}、b_i、ｃ

【００７３】

【数６】

$$\phi(x_0 + x_1 X + \cdots + x_{n-1} X^{n-1}) = (x_0, \cdots, x_{n-1})$$

…（６）

$$f : A \to A, X \mapsto \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} a_{ij} X^{q^i + q^j} + \sum_{i=0}^{n-1} b_i X^{q^i} + c$$

…（７）

【００７４】

但し、ｄをそれほど大きくない整数として、a_{ij}，b_i，$c \in A$、「$q^i + q^j > d$ならば$a_{ij} = 0$」かつ「$q^i > d$ならば$b_j = 0$」である。

【００７５】

≪ＨＦＥ関数Ｆｔの構造≫

ＨＦＥ関数F_tは、変換Ｓによる写像、中央写像Ｆ（$F = \phi^{-1} * f * \phi$）、及び変換Ｔによる写像の合成写像$F_t = T * F * S$により表現される（この＊は写像の合成）。そして、$y = F_t(x)$を計算するアルゴリズムは、次のようになる。

【００７６】

（Ｓｔｅｐ．１）変換Ｓにより、与えられた$x = (x_0, \cdots, x_{n-1}) \in K^n$を$x' = (x_0', \cdots, x_{n-1}') \in K^n$に変換する。

（Ｓｔｅｐ．２）ϕ^{-1}により、$x' \in K^n$を$X' \in A$に変換する。

（Ｓｔｅｐ．３）中央写像ｆにより、$X' \in A$を$Y' = f(X') \in A$に変換する。

（Ｓｔｅｐ．４）φにより、$Y' \in A$を$y' = (y_0', \cdots, y_{n-1}') \in K^n$に変換する。

（Ｓｔｅｐ．５）変換Ｔにより、$y' \in K^n$を$y = (y_0, \cdots, y_{n-1}) \in K^n$に変換する。

（Ｓｔｅｐ．６）$y \in K^n$を出力する。

【００７７】

上記の式（７）に示すように、ＨＦＥ関数$Ｆ_{t}$は、非線形の一変数多項式に基づく中央写像ｆを含んでいる。そのため、ある終域Ａの元Ｙ’に対し、一変数多項式の根の集合に相当する原像｛Ｚ∈Ａ｜ｆ（Ｚ）＝Ｙ’｝の要素数が複数存在する可能性がある。この場合、値域の元ｙに対し、ＨＦＥ関数$Ｆ_{t}$に関する原像の要素数が複数となる。

【００７８】

また、ある終域Ａの元Ｙ’に対し、原像｛Ｚ∈Ａ｜ｆ（Ｚ）＝Ｙ’｝の要素が全く存在しない可能性もある。この場合、原像｛Ｚ∈Ａ｜ｆ（Ｚ）＝Ｙ’｝の要素が全く存在しない終域の元は値域に含まれないため、終域と値域とは異なる。

【００７９】

以下、ＨＦＥ関数$Ｆ_{t}$の計算アルゴリズムについて、より詳細に説明する。

【００８０】

≪順方向の計算アルゴリズム≫

ＨＦＥ関数$Ｆ_{t}$に対する順方向の計算アルゴリズムは、与えられた$ｘ∈Ｋ^{n}$をＨＦＥ関数$Ｆ_{t}$（ｘ）に代入して$ｙ＝Ｆ_{t}$（ｘ）$∈Ｋ^{n}$を得るステップにより構成される。この順方向の計算アルゴリズムに定義域の元ｘが１つ入力されると値域の元ｙが１つ出力される。

【００８１】

≪逆方向の計算アルゴリズム≫

ＨＦＥ関数Ｆｔに関する逆方向の計算アルゴリズムは、次のＳｔｅｐ．１～Ｓｔｅｐ．７により構成される。

【００８２】

（Ｓｔｅｐ．１）与えられた$ｙ＝（ｙ_{0}，…，ｙ_{n－1}）∈Ｋ^{n}$を変換Ｔの逆変換$Ｔ^{－1}$に適用して$ｙ’＝（ｙ_{0}’，…，ｙ_{n－1}’）∈Ｋ^{n}$を得る。

（Ｓｔｅｐ．２）$φ^{－1}$により、$ｙ’＝（ｙ_{0}’，…，ｙ_{n－1}’）∈Ｋ^{n}$をＹ’∈Ａに変換する。

（Ｓｔｅｐ．３）Ｙ’を用いて、Ｘ’∈｛Ｚ∈Ａ｜ｆ（Ｚ）＝Ｙ’｝の集合を計算する。但し、｛Ｚ∈Ａ｜ｆ（Ｚ）＝Ｙ’｝が空集合の場合、例外値ｅｒｒを出力する。なお、Ｘ’∈｛Ｚ∈Ａ｜ｆ（Ｚ）＝Ｙ’｝は、例えば、多項式ｆ（Ｘ）－Ｙ’を因数分解することにより求められる。また、終域の要素Ｙ’をランダムに選択したとき、その要素Ｙ’に対する原像｛Ｚ∈Ａ｜ｆ（Ｚ）＝Ｙ’｝がｍ個の元を持つ確率は、近似的に１／（ｍ！ｅ）となる（但し、このｅはネイピア数）。

（Ｓｔｅｐ．４）Ｘ’∈｛Ｚ∈Ａ｜ｆ（Ｚ）＝Ｙ’｝の集合から１つの要素Ｘ’を選択する。

（Ｓｔｅｐ．５）φにより、Ｓｔｅｐ．４で選択された１つの要素Ｘ’∈Ａを$ｘ’＝（ｘ_{0}’，…，ｘ_{n－1}’）∈Ｋ^{n}$に変換する。

（Ｓｔｅｐ．６）変換Ｓの逆変換$Ｓ^{－1}$により、$ｘ’∈Ｋ^{n}$を$ｘ＝（ｘ_{0}，…，ｘ_{n－1}）∈Ｋ^{n}$に変換する。

（Ｓｔｅｐ．７）$ｘ∈Ｋ^{n}$を出力する。

【００８３】

上記のＳｔｅｐ．３において、Ｘ’∈｛Ｚ∈Ａ｜ｆ（Ｚ）＝Ｙ’｝の要素数α＝｜｛Ｚ∈Ａ｜ｆ（Ｚ）＝Ｙ’｝｜がα＝０又はα≧２となる可能性がある点に注意されたい。

【００８４】

（１－４－２：ＨＦＥ電子署名方式のアルゴリズム）

ここまでＨＦＥ関数$Ｆ_{t}$について説明してきた。次に、ＨＦＥ関数$Ｆ_{t}$を用いた電子署名方式（以下、ＨＦＥ署名方式）の具体的なアルゴリズムについて説明する。但し、ここでは、ＨＦＥ署名方式の一例として、ＨＦＥ関数を用いたＰＦＤＨ署名方式（以下、ＨＦＥ＋ＰＦＤＨ署名方式）について説明する。

【００８５】

≪ＰＦＤＨ署名方式≫

まず、ＰＦＤＨ署名方式における鍵生成アルゴリズムＧｅｎ、署名生成アルゴリズムＳ

ｉｇ、署名検証アルゴリズムＶｅｒについて説明する。これらＰＦＤＨ署名方式のアルゴリズムは、トラップドア付き一方向性関数F_t：Ａ→Ｂ、ハッシュ関数Ｈ：｛０，１｝＊→Ｂを利用する。

【００８６】

（鍵生成アルゴリズムＧｅｎ）

鍵生成アルゴリズムＧｅｎは、セキュリティパラメータを1^{λ}とし、署名鍵ｓｋをF_tのトラップドアｔとし、検証鍵ｐｋをF_tとして（ｓｋ，ｐｋ）を計算する（（ｓｋ，ｐｋ）←Ｇｅｎ（1^{λ}））。

【００８７】

（署名生成アルゴリズムＳｉｇ）

署名生成アルゴリズムＳｉｇは、メッセージＭ、署名鍵ｓｋを入力とし、次のＳｔｅｐ．１～Ｓｔｅｐ．４により電子署名σを計算する（σ←Ｓｉｇ（ｓｋ，Ｍ））。

【００８８】

（Ｓｔｅｐ．１）乱数ｒを生成する。

（Ｓｔｅｐ．２）ｙ＝Ｈ（Ｍ，ｒ）∈Ｂを計算する。

（Ｓｔｅｐ．３）トラップドアｔを用いてF_tの逆方向の計算アルゴリズムを実行し、ｙ＝F_t（ｘ）となるｘを算出する。但し、ｙ＝F_t（ｘ）となるｘが存在しない場合にはＳｔｅｐ．１に戻る。

（Ｓｔｅｐ．４）電子署名σ＝（ｘ，ｒ）を出力する。

【００８９】

（署名検証アルゴリズムＶｅｒ）

署名検証アルゴリズムＶｅｒは、検証鍵ｐｋ＝F_t、メッセージＭ、電子署名σ＝（ｘ，ｒ）を入力とし、次のＳｔｅｐ．１及びＳｔｅｐ．２によりメッセージＭに対する電子署名σの正当性を検証する（０／１←Ｖｅｒ（ｐｋ，Ｍ，σ））。

【００９０】

（Ｓｔｅｐ．１）F_t（ｘ）＝Ｈ（Ｍ，ｒ）か否かを判断する。

（Ｓｔｅｐ．２）F_t（ｘ）＝Ｈ（Ｍ，ｒ）の場合に１を出力し、F_t（ｘ）≠Ｈ（Ｍ，ｒ）の場合に０を出力する。

【００９１】

≪ＨＦＥ＋ＰＦＤＨ署名方式≫

次に、ＨＦＥ＋ＰＦＤＨ署名方式における署名生成アルゴリズムＳｉｇ、署名検証アルゴリズムＶｅｒについて説明する。ＨＦＥ＋ＰＦＤＨ署名方式は、ＨＦＥ関数F_tを用いたＰＦＤＨ署名方式である。なお、ＨＦＥ＋ＰＦＤＨ署名方式では、署名鍵ｓｋ、ＨＦＥ関数F_tのトラップドアＳ，Ｔ，a_{ij}、b_i、ｃ、検証鍵ｐｋがF_tに設定される。

【００９２】

（署名生成アルゴリズムＳｉｇ）

署名生成アルゴリズムＳｉｇは、メッセージＭ、署名鍵ｓｋを入力とし、次のＳｔｅｐ．１～Ｓｔｅｐ．９により電子署名σを計算する（σ←Ｓｉｇ（ｓｋ，Ｍ））。

【００９３】

（Ｓｔｅｐ．１）乱数ｒを生成する。

（Ｓｔｅｐ．２）乱数ｒ、メッセージＭを用いて、ハッシュ値ｙ∈K^n←Ｈ（Ｍ，ｒ）を計算する。

（Ｓｔｅｐ．３）ｙ＝（y_0，…，y_{n-1}）∈K^nを変換Ｔの逆変換T^{-1}に適用してｙ’＝（y_0’，…，y_{n-1}’）∈K^nを得る。

（Ｓｔｅｐ．４）ϕ^{-1}により、ｙ’＝（y_0’，…，y_{n-1}’）∈K^nをＹ’∈Ａに変換する。

（Ｓｔｅｐ．５）Ｘ’∈｛Ｚ∈Ａ｜ｆ（Ｚ）＝Ｙ’｝の集合を計算する。

（Ｓｔｅｐ．６）集合｛Ｚ∈Ａ｜ｆ（Ｚ）＝Ｙ’｝から１つの要素Ｘ’を選択する。但し、集合｛Ｚ∈Ａ｜ｆ（Ｚ）＝Ｙ’｝が空集合の場合、Ｓｔｅｐ．１の処理へ戻る。

（Ｓｔｅｐ．７）φにより、Ｘ’∈Ａをｘ’＝（x_0’，…，x_{n-1}’）∈K^nに変

換する。
（Ｓｔｅｐ．８）変換Ｓにより、ｘ’∈Ｋ^{n}をｘ＝（ｘ$_{0}$，…，ｘ$_{n-1}$）∈Ｋ^{n}に変換する。
（Ｓｔｅｐ．９）電子署名σ＝（ｘ，ｒ）を出力する。
【００９４】
（署名検証アルゴリズムＶｅｒ）
署名検証アルゴリズムＶｅｒは、検証鍵ｐｋ＝Ｆ$_{t}$、メッセージＭ、電子署名σ＝（ｘ，ｒ）を入力とし、次のＳｔｅｐ．１～Ｓｔｅｐ．３によりメッセージＭに対する電子署名σの正当性を検証する（０／１←Ｖｅｒ（ｐｋ，Ｍ，σ））。
【００９５】
（Ｓｔｅｐ．１）電子署名σに含まれるｒ、及びメッセージＭを用いて、ハッシュ値ｙ←Ｈ（Ｍ，ｒ）を計算する。
（Ｓｔｅｐ．２）電子署名σに含まれるｘ∈Ｋ^{n}をＨＦＥ関数Ｆ$_{t}$（ｘ）に代入して、ｙ”＝Ｆ$_{t}$（ｘ）∈Ｋ^{n}を計算する。
（Ｓｔｅｐ．３）ｙ＝ｙ”ならば１を出力し、ｙ≠ｙ”ならば０を出力する。
【００９６】
以上、ＨＦＥ電子署名方式（ここではＨＦＥ＋ＰＦＤＨ署名方式を例示した。）における各アルゴリズムの構成について説明した。上記のアルゴリズム構成から推察される通り、ＨＦＥ電子署名方式においては、Ｈ（Ｍ，ｒ）に関するＦ$_{t}$＝Ｔ＊Ｆ＊Ｓの原像が計算されると電子署名σが偽造されてしまう。つまり、ＨＦＥ電子署名方式は、Ｈ（Ｍ，ｒ）＝Ｆ$_{t}$（ｘ）で表現される特殊な多次多変数連立方程式の解答困難性に依存した方式である。そして、このような特殊な多次多変数連立方程式については、その求解問題の困難性に対する評価が難しい。実際には、ＨＦＥ電子署名方式で用いられる多次多変数連立方程式は、準指数時間（ｑｕａｓｉ－ｐｏｌｙｎｏｍｉｎａｌ　ｔｉｍｅ）以下で解くことができると言われている。
【００９７】
＜２：第１実施形態＞
ここで、本発明の第１実施形態について説明する。本実施形態は、ＨＦＥ電子署名方式と同様、多次多変数連立方程式に対する求解問題の困難性に安全性の根拠をおく公開鍵認証方式及び電子署名方式に関する。但し、本実施形態は、ＨＦＥ電子署名方式などとは異なり、十分な安全性を確保するために、効率的に解く手段（トラップドア）を持たない多次多変数連立方程式を利用できるようにした公開鍵認証方式及び電子署名方式を提供するものである。
【００９８】
［２－１：公開鍵認証方式のアルゴリズム］
まず、図４を参照しながら、本実施形態に係る公開鍵認証方式（以下、本手法）のアルゴリズムについて説明する。図４は、本手法のアルゴリズムについて説明するための説明図である。なお、本手法は、鍵生成アルゴリズムＧｅｎ、証明者アルゴリズムＰ、検証者アルゴリズムＶにより構成される。以下、各アルゴリズムの内容について説明する。
【００９９】
（鍵生成アルゴリズムＧｅｎ）
鍵生成アルゴリズムＧｅｎは、環Ｋ上で定義されるｍ本のｎ変数多次多項式ｆ$_{1}$（ｘ$_{1}$，…，ｘ$_{n}$），…，ｆ$_{m}$（ｘ$_{1}$，…，ｘ$_{n}$）と、ベクトルｓ＝（ｓ$_{1}$，…，ｓ$_{n}$）∈Ｋ^{n}を生成する。次に、鍵生成アルゴリズムＧｅｎは、ｙ＝（ｙ$_{1}$，…，ｙ$_{m}$）←（ｆ$_{1}$（ｓ），…，ｆ$_{m}$（ｓ））を計算する。そして、鍵生成アルゴリズムＧｅｎは、（ｆ$_{1}$，…，ｆ$_{m}$，ｙ）を公開鍵ｐｋに設定し、ｓを秘密鍵に設定する。なお、以下では、ｎ変数のベクトル（ｘ$_{1}$，…，ｘ$_{n}$）をｘと表記し、ｍ本のｎ変数多次多項式（ｆ$_{1}$（ｘ），…，ｆ$_{m}$（ｘ））をＦ（ｘ）と表記する。
【０１００】
（証明者アルゴリズムＰ、検証者アルゴリズムＶ）

次に、図４を参照しながら、対話プロトコルにおける証明者アルゴリズムＰと検証者アルゴリズムＶによる処理について説明する。この対話プロトコルは、「証明者がｙ＝Ｆ（ｓ）を満たすｓを知っていること」をｓの情報を検証者に一切漏らさずに、検証者に証明させるものである。なお、鍵生成アルゴリズムＧｅｎにより生成された公開鍵ｐｋは、証明者と検証者の間で共有されているものとする。また、鍵生成アルゴリズムＧｅｎにより生成された秘密鍵ｓは、証明者により秘密に管理されているものとする。

【０１０１】

Ｓｔｅｐ．１：

まず、証明者アルゴリズムＰは、任意に数ｗを選択する。次いで、証明者アルゴリズムＰは、擬似乱数生成器G_1に数ｗを適用してベクトルｒ∈K^nと数ｗ’を生成する。つまり、証明者アルゴリズムＰは、（ｒ，ｗ’）←G_1（ｗ）を計算する。次いで、証明者アルゴリズムＰは、擬似乱数生成器G_2に数ｗ’を適用してｎ変数多項式Ｆ’（ｘ）＝（f'_1（ｘ），…，f'_m（ｘ））を生成する。つまり、証明者アルゴリズムＰは、Ｆ’←G_2（ｗ’）を計算する。

【０１０２】

Ｓｔｅｐ．１（続き）：

次いで、証明者アルゴリズムＰは、ｚ←ｓ－ｒを計算する。この計算は、秘密鍵ｓをベクトルｒによりマスクする操作に相当する。さらに、証明者アルゴリズムＰは、Ｆ”（ｘ）←Ｆ（ｘ＋ｒ）＋Ｆ’（ｘ）を計算する。この計算は、ｘについての多項式の組Ｆ（ｘ＋ｒ）を多項式の組Ｆ’（ｘ）によりマスクする操作に相当する。

【０１０３】

Ｓｔｅｐ．１（続き）：

次いで、証明者アルゴリズムＰは、Ｆ’（ｚ）とｚのハッシュ値c_1を生成する。つまり、証明者アルゴリズムＰは、c_1←H_1（Ｆ’（ｚ），ｚ）を計算する。また、証明者アルゴリズムＰは、数ｗ’のハッシュ値c_2を生成する。つまり、証明者アルゴリズムＰは、c_2←H_2（ｗ’）を計算する。さらに、証明者アルゴリズムＰは、多項式の組Ｆ”のハッシュ値c_3を生成する。つまり、証明者アルゴリズムＰは、c_3←H_3（Ｆ”）を計算する。なお、上記のH_1（…）、H_2（…）、H_3（…）は、ハッシュ関数である。また、ハッシュ値（c_1，c_2，c_3）はメッセージである。

【０１０４】

Ｓｔｅｐ．１で生成されたメッセージ（c_1，c_2，c_3）は、検証者に送られる。

【０１０５】

Ｓｔｅｐ．２：

検証者アルゴリズムＶは、３つの検証パターンのうち、どの検証パターンを利用するかを選択する。次いで、検証者アルゴリズムＶは、選択した検証パターンを表す要求ｄ∈｛０，１，２｝を証明者に送る。

【０１０６】

Ｓｔｅｐ．３：

証明者アルゴリズムＰは、検証者から受けた要求ｄに応じて検証者に送り返す情報σを生成する。もしｄ＝０の場合、証明者アルゴリズムＰは、情報σ＝ｗを生成する。また、ｄ＝１の場合、証明者アルゴリズムＰは、情報σ＝（ｗ’，ｚ）を生成する。そして、ｄ＝２の場合、証明者アルゴリズムＰは、情報σ＝（Ｆ”（ｘ），ｚ）を生成する。このようにして生成された情報σは、証明者アルゴリズムＰにより検証者に送られる。

【０１０７】

Ｓｔｅｐ．４：

検証者アルゴリズムＶは、証明者から受け取った情報σを利用して以下の検証処理を実行する。

【０１０８】

ｄ＝０の場合、検証者アルゴリズムＶは、（ｒ’，ｗ”）←G_1（σ）を計算する。さらに、検証者アルゴリズムＶは、Ｆ’’’←G_2（ｗ”）を計算する。そして、検証者アル

ゴリズムＶは、$c_2 = H_2$（ｗ”）の等号が成り立つか否かを検証する。また、検証者アルゴリズムＶは、$c_3 = H_3$（Ｆ（ｘ＋ｒ’）＋Ｆ'''（ｘ））の等号が成り立つか否かを検証する。検証者アルゴリズムＶは、これらの検証が全て成功した場合に認証成功を示す値１を出力し、検証に失敗があった場合に認証失敗を示す値０を出力する。

【０１０９】

ｄ＝１の場合、検証者アルゴリズムＶは、（ｗ”，ｚ’）←σとする。さらに、検証者アルゴリズムＶは、$F''' \leftarrow G_2$（ｗ”）を計算する。そして、検証者アルゴリズムＶは、$c_1 = H_1$（Ｆ'''（ｚ’），ｚ’）の等号が成り立つか否かを検証する。また、検証者アルゴリズムＶは、$c_2 = H_2$（ｗ”）の等号が成り立つか否かを検証する。検証者アルゴリズムＶは、これらの検証が全て成功した場合に認証成功を示す値１を出力し、検証に失敗があった場合に認証失敗を示す値０を出力する。

【０１１０】

ｄ＝２の場合、検証者アルゴリズムＶは、（Ｆ''''，ｚ’）←σとする。そして、検証者アルゴリズムＶは、$c_1 = H_1$（Ｆ''''（ｚ’）－ｙ，ｚ’））の等号が成り立つか否かを検証する。さらに、検証者アルゴリズムＶは、$c_3 = H_3$（Ｆ''''）の等号が成り立つか否かを検証する。検証者アルゴリズムＶは、これらの検証が全て成功した場合に認証成功を示す値１を出力し、検証に失敗があった場合に認証失敗を示す値０を出力する。

【０１１１】

（補足）

さて、上記Ｓｔｅｐ．１において生成されたメッセージ（c_1，c_2，c_3）を検証者に送った際、秘密鍵ｓｋに関する情報、ｒに関する情報、ｚに関する情報が検証者に一切漏れていないことに注意されたい。また、上記Ｓｔｅｐ．３において生成された情報σを検証者に送った際、ｄ＝０のときにはｚに関する情報が検証者に一切漏れておらず、ｄ＝１，２のときにはｒに関する情報が検証者に一切漏れていないことに注意されたい。

【０１１２】

（本手法における健全性について）

本手法の健全性は、検証者の全ての要求ｄ＝０，１，２に対して、証明者がメッセージ（c_1，c_2，c_3）に対する正しい情報σを回答したならば、その回答内容から、下記の式（８）及び式（９）を満たすＦ''''、Ｆ'''、ｒ’、ｚ’が計算できるということから保証されている。

【０１１３】

【数７】

$$F''''(x) = F(x + r') + F'''(x)$$

…（８）

$$F''''(z') - y = F'''(z')$$

…（９）

【０１１４】

このようなロジックが保証されることにより、多次多変数連立方程式の求解問題を解かない限りにおいて２／３より高い確率で偽証することは不可能であることが保証される。つまり、検証者の要求ｄ＝０，１，２の全てに正しく回答するためには、偽証者が上記の式（８）及び式（９）を満たすＦ''''、Ｆ'''、ｒ’、ｚ’を計算できる必要がある。言い換えると、偽証者はＦ（ｓ）＝ｙを満たすｓを計算できる必要がある。なお、検証者の要求ｄ＝０，１，２の高々２つまでに偽証者が正しく回答する可能性はある。そのため、

偽証の成功確率は２／３となる。上記の対話プロトコルは十分な回数実行される。そのため、偽証の成功確率は無視できるほど小さくすることができる。

【０１１５】

（変形例）

上記の鍵生成アルゴリズムＧｅｎは、ｙ←Ｆ（ｓ）を計算し、公開鍵を（Ｆ，ｙ）に設定している。しかし、鍵生成アルゴリズムＧｅｎは、（y_1，…，y_m）←Ｆ（ｓ）として、（f_1^*（ｘ），…，f_m^*（ｘ））←（f_1（ｘ）－y_1，…，f_m（ｘ）－y_m）を計算し、公開鍵を（f_1^*，…，f_m^*）に設定するように構成されていてもよい。このように変形すると、証明者と検証者との間でｙ＝０として対話プロトコルを実行することが可能になる。

【０１１６】

また、上記の証明者アルゴリズムＰは、Ｆ”（ｚ）とｚからメッセージc_1を生成しているが、Ｆ”（ｚ）＝Ｆ’（ｚ）の関係があることにより、Ｆ’（ｚ）とｚからメッセージc_1を生成しても、同様の対話プロトコルを構成することができる。また、証明者アルゴリズムＰは、Ｆ”（ｚ）のハッシュ値と、ｚのハッシュ値とを別々に計算し、それぞれをメッセージとして検証者に送るようにしてもよい。

【０１１７】

また、上記の証明者アルゴリズムＰは、数ｗに擬似乱数生成器G_1を適用してベクトルｒと数ｗ’を生成している。そして、上記の証明者アルゴリズムＰは、数ｗ’を擬似乱数生成器G_2に適用してｎ変数多項式Ｆ’（ｘ）を生成している。しかし、証明者アルゴリズムＰは、最初からｗ＝（ｒ，Ｆ’）を算出し、G_1を恒等写像としてもよい。また、この場合には、数ｗをG_1に適用する必要はない。なお、G_2についても同様である。

【０１１８】

また、上記の対話プロトコルにおいては公開鍵を（Ｆ，ｙ）としている。このＦは秘密鍵ｓｋに依らないパラメータである。そのため、このパラメータＦを証明者毎に設けるのではなく、このパラメータＦをシステム全体に共通するパラメータにしてもよい。この場合、証明者毎に設定される公開鍵はｙだけで済むことになり、公開鍵のサイズを小さくすることが可能になる。

【０１１９】

また、上記の対話プロトコルにおいては公開鍵を（f_1，…，f_m，ｙ）としているが、Ｆ＝（f_1，…，f_m）は適当に選択してもよいパラメータである。そのため、証明者と検証者は、（f_1，…，f_m）をそのまま保持するのではなく、例えば、乱数のシードw_{pk}を用意し、擬似乱数生成器G^*を利用してＦ←G^*（w_{pk}）を計算するようにしてもよい。この場合、公開鍵は（w_{pk}，ｙ）となり、（Ｆ，ｙ）を公開鍵として公開する場合よりも、公開鍵のサイズを小さくすることが可能になる。

【０１２０】

上記の方式では、ハッシュ関数H_1，H_2，H_3を用いてc_1，c_2，c_3を計算しているが、ハッシュ関数の代わりにコミットメント方式ＣＯＭを用いてもよい。コミットメント関数ＣＯＭは、文字列Ｓと乱数ρの２つを引数にとる関数である。コミットメント関数の例としては、Ｓｈａｉ　ＨａｌｅｖｉとＳｉｌｖｉｏ　Ｍｉｃａｌｉによって国際会議ＣＲＹＰＴＯ１９９６で発表された方式などがある。

【０１２１】

このコミットメント関数を用いる場合、c_1，c_2，c_3を計算する前に、乱数ρ_1，ρ_2，ρ_3を用意し、ハッシュ関数H_1（・），H_2（・），H_3（・）を適用する代わりにコミットメント関数ＣＯＭ（・，ρ_1），ＣＯＭ（・，ρ_2），ＣＯＭ（・，ρ_2）を適用することによりc_1，c_2，c_3を生成する。また、この変形例では、検証部で計算するC_iを生成するために必要なρ_iを回答情報に含めることになる。なお、この変形手法は、後述の全ての方式について適用可能である。

【０１２２】

以上、本手法に係る基本的なアルゴリズム構成について説明した。

【０１２３】
［２－２：拡張アルゴリズム］
次に、図５を参照しながら、本手法を拡張した公開鍵認証方式（以下、拡張手法）のアルゴリズムについて説明する。図５は、拡張手法に基づく対話プロトコルの流れを説明するための説明図である。この拡張手法は、１パス目に送信するメッセージ（c_1，c_2，c_3）を１つのハッシュ値ｃに変換して検証者に送る方式である。また、この拡張手法では、１パス目にハッシュ値ｃを送るように対話プロトコルを構成するため、３パス目で送る情報σから復元できないメッセージを情報σと共に検証者に送る。このように拡張することで、対話プロトコルの中で検証者に送るハッシュ値の数を減らすことが可能になり、通信するデータサイズを削減することができる。以下、拡張方式における各アルゴリズムの内容について詳細に説明する。
【０１２４】
（鍵生成アルゴリズムＧｅｎ）
鍵生成アルゴリズムＧｅｎは、環Ｋ上で定義されるｍ本のｎ変数多次多項式$f_1(x_1, \cdots, x_n), \cdots, f_m(x_1, \cdots, x_n)$と、ベクトル$s=(s_1, \cdots, s_n) \in K^n$を生成する。次に、鍵生成アルゴリズムＧｅｎは、$y=(y_1, \cdots, y_m) \leftarrow (f_1(s), \cdots, f_m(s))$を計算する。そして、鍵生成アルゴリズムＧｅｎは、$(f_1, \cdots, f_m, y)$を公開鍵ｐｋに設定し、ｓを秘密鍵に設定する。なお、以下では、ｎ変数のベクトル$(x_1, \cdots, x_n)$をｘと表記し、ｍ本のｎ変数多次多項式$(f_1(x), \cdots, f_m(x))$をＦ（ｘ）と表記する。
【０１２５】
（証明者アルゴリズムＰ、検証者アルゴリズムＶ）
次に、図５を参照しながら、対話プロトコルにおける証明者アルゴリズムＰと検証者アルゴリズムＶによる処理について説明する。この対話プロトコルは、「証明者が$y=F(s)$を満たすｓを知っていること」をｓの情報を検証者に一切漏らさずに、検証者に証明させるものである。なお、鍵生成アルゴリズムＧｅｎにより生成された公開鍵ｐｋは、証明者と検証者の間で共有されているものとする。また、鍵生成アルゴリズムＧｅｎにより生成された秘密鍵ｓは、証明者により秘密に管理されているものとする。
【０１２６】
Ｓｔｅｐ．１：
まず、証明者アルゴリズムＰは、任意に数ｗを選択する。次いで、証明者アルゴリズムＰは、擬似乱数生成器Ｇ１に数ｗを適用してベクトル$r \in K^n$と数ｗ’を生成する。つまり、証明者アルゴリズムＰは、$(r, w') \leftarrow G_1(w)$を計算する。次いで、証明者アルゴリズムＰは、擬似乱数生成器G_2に数ｗ’を適用してｎ変数多項式$F'(x)=(f'_1(x), \cdots, f'_m(x))$を生成する。つまり、証明者アルゴリズムＰは、$F' \leftarrow G_2(w')$を計算する。
【０１２７】
Ｓｔｅｐ．１（続き）：
次いで、証明者アルゴリズムＰは、$z \leftarrow s-r$を計算する。この計算は、秘密鍵ｓをベクトルｒによりマスクする操作に相当する。さらに、証明者アルゴリズムＰは、$F''(x) \leftarrow F(x+r)+F'(x)$を計算する。この計算は、ｘについての多項式の組Ｆ（ｘ＋ｒ）を多項式の組Ｆ’（ｘ）によりマスクする操作に相当する。
【０１２８】
Ｓｔｅｐ．１（続き）：
次いで、証明者アルゴリズムＰは、Ｆ”（ｚ）とｚのハッシュ値c_1を生成する。つまり、証明者アルゴリズムＰは、$c_1 \leftarrow H_1(F''(z), z)$を計算する。また、証明者アルゴリズムＰは、数ｗ’のハッシュ値c_2を生成する。つまり、証明者アルゴリズムＰは、$c_2 \leftarrow H_2(w')$を計算する。さらに、証明者アルゴリズムＰは、多項式の組Ｆ”のハッシュ値c_3を生成する。つまり、証明者アルゴリズムＰは、$c_3 \leftarrow H_3(F'')$を計算する。なお、上記の$H_1(\cdots)$、$H_2(\cdots)$、$H_3(\cdots)$は、ハッシュ関数である。

また、ハッシュ値（c_1，c_2，c_3）はメッセージである。

【０１２９】

Ｓｔｅｐ．１（続き）：

拡張方式の場合、証明者アルゴリズムＰは、メッセージ（c_1，c_2，c_3）をハッシュ関数Ｈに適用してハッシュ値ｃを生成する。そして、証明者アルゴリズムＰは、生成したハッシュ値ｃを検証者に送る。

【０１３０】

Ｓｔｅｐ．２：

検証者アルゴリズムＶは、３つの検証パターンのうち、どの検証パターンを利用するかを選択する。次いで、検証者アルゴリズムＶは、選択した検証パターンを表す要求ｄ∈｛０，１，２｝を証明者に送る。

【０１３１】

Ｓｔｅｐ．３：

証明者アルゴリズムＰは、検証者から受けた要求ｄに応じて検証者に送り返す情報σを生成する。もしｄ＝０の場合、証明者アルゴリズムＰは、情報（σ，c^*）＝（ｗ，c_1）を生成する。また、ｄ＝１の場合、証明者アルゴリズムＰは、情報（σ，c^*）＝（（ｗ’，ｚ），c_3）を生成する。そして、ｄ＝２の場合、証明者アルゴリズムＰは、情報（σ，c^*）＝（（Ｆ”，ｚ），c_2）を生成する。このようにして生成された情報（σ，c^*）は、証明者アルゴリズムＰにより検証者に送られる。

【０１３２】

Ｓｔｅｐ．４：

検証者アルゴリズムＶは、証明者から受け取った情報（σ，c^*）を利用して以下の検証処理を実行する。

【０１３３】

ｄ＝０の場合、検証者アルゴリズムＶは、（ｒ’，ｗ”）←G_1（σ）を計算する。次いで、検証者アルゴリズムＶは、Ｆ'''←G_2（ｗ”）を計算する。次いで、検証者アルゴリズムＶは、c_2’＝H_2（ｗ”）を計算する。次いで、検証者アルゴリズムＶは、c_3’＝H_3（Ｆ（ｘ＋ｒ’）＋Ｆ'''（ｘ））を計算する。その後、検証者アルゴリズムＶは、ｃ＝Ｈ（c^*，c_2’，c_3’）の等号が成り立つか否かを検証する。そして、検証者アルゴリズムＶは、検証が成功した場合に認証成功を示す値１を出力し、検証が失敗した場合に認証失敗を示す値０を出力する。

【０１３４】

ｄ＝１の場合、検証者アルゴリズムＶは、（ｗ”，ｚ’）←σとする。次いで、検証者アルゴリズムＶは、Ｆ'''←G_2（ｗ”）を計算する。次いで、検証者アルゴリズムＶは、c_1’＝H_1（Ｆ'''（ｚ’），ｚ’）を計算する。次いで、検証者アルゴリズムＶは、c_2’＝H_2（ｗ”）を計算する。その後、検証者アルゴリズムＶは、ｃ＝Ｈ（c_1’，c_2’，c^*）の等号が成り立つか否かを検証する。そして、検証者アルゴリズムＶは、検証が成功した場合に認証成功を示す値１を出力し、検証が失敗した場合に認証失敗を示す値０を出力する。

【０１３５】

ｄ＝２の場合、検証者アルゴリズムＶは、（Ｆ''''，ｚ’）←σとする。次いで、検証者アルゴリズムＶは、c_1’＝H_1（Ｆ''''（ｚ’）－ｙ，ｚ’）を計算する。次いで、検証者アルゴリズムＶは、c_3’＝H_3（Ｆ''''）を計算する。その後、検証者アルゴリズムＶは、ｃ＝Ｈ（c_1’，c^*，c_3’）の等号が成り立つか否かを検証する。そして、検証者アルゴリズムＶは、検証が成功した場合に認証成功を示す値１を出力し、検証が失敗した場合に認証失敗を示す値０を出力する。

【０１３６】

以上、拡張方式の対話プロトコルにおける各アルゴリズムの処理について説明した。このように拡張することで、対話プロトコルの中で検証者に送るハッシュ値の数を減らすことが可能になり、通信するデータサイズを削減することができる。

【０１３７】
［２－３：並列化アルゴリズム］
さて、先に述べた通り、本手法又は拡張手法に係る対話プロトコルを適用すれば、偽証が成功する確率を２／３以下に抑制することができる。従って、この対話プロトコルを２回実行すれば、偽証が成功する確率を（２／３）^{2}以下に抑制することができる。同様に、この対話プロトコルをＮ回実行すると、偽証が成功する確率は（２／３）^{N}となり、Ｎを十分に大きい数（例えば、Ｎ＝１４０）にすれば、偽証が成功する確率は無視できる程度に小さくなる。例えば、本手法に係る対話プロトコルを並列的にＮ回実行するアルゴリズムは図６のようになる。以下、図６を参照しながら、並列的に対話プロトコルをＮ回実行する各アルゴリズムの内容について説明する。
【０１３８】
（鍵生成アルゴリズムＧｅｎ）
鍵生成アルゴリズムＧｅｎは、環Ｋ上で定義されるｍ本のｎ変数多次多項式$f_1(x_1, \cdots, x_n), \cdots, f_m(x_1, \cdots, x_n)$と、ベクトル$s=(s_1, \cdots, s_n) \in K^n$を生成する。次に、鍵生成アルゴリズムＧｅｎは、$y=(y_1, \cdots, y_m) \leftarrow (f_1(s), \cdots, f_m(s))$を計算する。そして、鍵生成アルゴリズムＧｅｎは、$(f_1, \cdots, f_m, y)$を公開鍵ｐｋに設定し、ｓを秘密鍵に設定する。なお、以下では、ｎ変数のベクトル$(x_1, \cdots, x_n)$をｘと表記し、ｍ本のｎ変数多次多項式$(f_1(x), \cdots, f_m(x))$をＦ（ｘ）と表記する。
【０１３９】
（証明者アルゴリズムＰ、検証者アルゴリズムＶ）
次に、図６を参照しながら、対話プロトコルにおける証明者アルゴリズムＰと検証者アルゴリズムＶによる処理について説明する。この対話プロトコルは、「証明者がｙ＝Ｆ（ｓ）を満たすｓを知っていること」をｓの情報を検証者に一切漏らさずに、検証者に証明させるものである。なお、鍵生成アルゴリズムＧｅｎにより生成された公開鍵ｐｋは、証明者と検証者の間で共有されているものとする。また、鍵生成アルゴリズムＧｅｎにより生成された秘密鍵ｓは、証明者により秘密に管理されているものとする。
【０１４０】
Ｓｔｅｐ．１：
まず、証明者アルゴリズムＰは、ｉ＝１～Ｎについて、以下の処理（１）～処理（８）を実行する。（処理（１））証明者アルゴリズムＰは、任意に数w_iを選択する。（処理（２））証明者アルゴリズムＰは、擬似乱数生成器G_1に数w_iを適用してベクトル$r_i \in K^n$と数w'_iを生成する。つまり、証明者アルゴリズムＰは、$(r_i, w'_i) \leftarrow G_1(w_i)$を計算する。（処理（３））証明者アルゴリズムＰは、擬似乱数生成器G_2に数w'_iを適用してｎ変数多項式の組$F'_i(x)$を生成する。つまり、証明者アルゴリズムＰは、$F'_i \leftarrow G_2(w'_i)$を計算する。
【０１４１】
Ｓｔｅｐ．１（続き）：
（処理（４））証明者アルゴリズムＰは、$z_i \leftarrow s_i - r_i$を計算する。この計算は、秘密鍵s_iをベクトルr_iによりマスクする操作に相当する。（処理（５））証明者アルゴリズムＰは、$F''_i(x) \leftarrow F(x+r_i) + F'_i(x)$を計算する。この計算は、ｘについての多項式の組$F(x+r_i)$を多項式の組$F'_i(x)$によりマスクする操作に相当する。
【０１４２】
Ｓｔｅｐ．１（続き）：
（処理（６））証明者アルゴリズムＰは、$F''_i(z_i)$とz_iのハッシュ値$c_{1,i}$を生成する。つまり、証明者アルゴリズムＰは、$c_{1,i} \leftarrow H_1(F''_i(z_i), z_i)$を計算する。（処理（７））証明者アルゴリズムＰは、数w'_iのハッシュ値$c_{2,i}$を生成する。つまり、証明者アルゴリズムＰは、$c_{2,i} \leftarrow H_2(w'_i)$を計算する。（処理（８））証明者アルゴリズムＰは、多項式の組F''_iのハッシュ値$c_{3,i}$を生成

する。つまり、証明者アルゴリズムＰは、$c_{3,i} \leftarrow H_3(F''_i)$を計算する。なお、上記の$H_1(\cdots)$、$H_2(\cdots)$、$H_3(\cdots)$は、ハッシュ関数である。また、ハッシュ値（$c_{1,i}$、$c_{2,i}$、$c_{3,i}$）はメッセージである。

【０１４３】

ｉ＝１～Ｎについて、上記の（処理（１））～（処理（８））が実行された後、Ｓｔｅｐ．１で生成されたメッセージ（$c_{1,i}$，$c_{2,i}$，$c_{3,i}$）（ｉ＝１～Ｎ）は、検証者に送られる。

【０１４４】

Ｓｔｅｐ．２：

検証者アルゴリズムＶは、ｉ＝１～Ｎのそれぞれについて、３つの検証パターンのうち、どの検証パターンを利用するかを選択する。次いで、検証者アルゴリズムＶは、選択した検証パターンを表す要求$d_i \in \{0, 1, 2\}$（ｉ＝１～Ｎ）を証明者に送る。

【０１４５】

Ｓｔｅｐ．３：

証明者アルゴリズムＰは、検証者から受けた要求d_iに応じて検証者に送り返す情報σ_iを生成する。ここで、証明者アルゴリズムＰは、ｉ＝１～Ｎについて、以下の処理（１）～処理（３）を実行する。（処理（１））$d_i=0$の場合、証明者アルゴリズムＰは、情報$\sigma_i = w_i$を生成する。（処理（２））$d_i=1$の場合、証明者アルゴリズムＰは、情報$\sigma_i = (w'_i, z_i)$を生成する。（処理（３））$d_i=2$の場合、証明者アルゴリズムＰは、情報$\sigma_i = (F''_i, z_i)$を生成する。上記の処理（１）～（３）の判定及び処理が実行された後、情報σ_i（ｉ＝１～Ｎ）は、証明者アルゴリズムＰにより検証者に送られる。

【０１４６】

Ｓｔｅｐ．４：

検証者アルゴリズムＶは、証明者から受け取った情報σ_i（ｉ＝１～Ｎ）を利用して以下の検証処理を実行する。なお、以下の処理は、ｉ＝１～Ｎについて実行される。

【０１４７】

$d_i=0$の場合、検証者アルゴリズムＶは、$(r'_i, w''_i) \leftarrow G_1(\sigma_i)$を計算する。さらに、検証者アルゴリズムＶは、$F'''_i \leftarrow G_2(w''_i)$を計算する。そして、検証者アルゴリズムＶは、$c_{2,i} = H_2(w''_i)$の等号が成り立つか否かを検証する。また、検証者アルゴリズムＶは、$c_{3,i} = H_3(F(x+r'_i)+F'''_i(x))$の等号が成り立つか否かを検証する。検証者アルゴリズムＶは、これらの検証が全て成功した場合に認証成功を示す値１を出力し、検証に失敗があった場合に認証失敗を示す値０を出力する。

【０１４８】

$d_i=1$の場合、検証者アルゴリズムＶは、$(w''_i, z'_i) \leftarrow \sigma_i$とする。さらに、検証者アルゴリズムＶは、$F'''_i \leftarrow G_2(w''_i)$を計算する。そして、検証者アルゴリズムＶは、$c_{1,i} = H_1(F'''_i(z'_i), z'_i)$の等号が成り立つか否かを検証する。また、検証者アルゴリズムＶは、$c_2 = H_2(w''_i)$の等号が成り立つか否かを検証する。検証者アルゴリズムＶは、これらの検証が全て成功した場合に認証成功を示す値１を出力し、検証に失敗があった場合に認証失敗を示す値０を出力する。

【０１４９】

$d_i=2$の場合、検証者アルゴリズムＶは、$(F''''_i, z'_i) \leftarrow \sigma_i$とする。そして、検証者アルゴリズムＶは、$c_{1,i} = H_1(F''''_i(z'_i)-y, z'_i)$の等号が成り立つか否かを検証する。さらに、検証者アルゴリズムＶは、$c_{3,i} = H_3(F''''_i(x))$の等号が成り立つか否かを検証する。検証者アルゴリズムＶは、これらの検証が全て成功した場合に認証成功を示す値１を出力し、検証に失敗があった場合に認証失敗を示す値０を出力する。

【０１５０】

以上、本手法の対話プロトコルを並列的に実行する方法について説明した。上記のよう

に、本手法の対話プロトコルを繰り返し実行することにより、偽証が成功する確率を無視できる程度まで低減させることが可能になる。

【０１５１】

なお、上記のＳｔｅｐ．１の後、（$c_{1,1}$，$c_{1,2}$，$c_{1,3}$，…，$c_{N,1}$，$c_{N,2}$，$c_{N,3}$）を検証者に送信する代わりに，これらをハッシュ値Ｈ（$c_{1,1}$，$c_{1,2}$，$c_{1,3}$，…，$c_{N,1}$，$c_{N,2}$，$c_{N,3}$）にまとめて送信してもよい。この構成を適用すると、１パス目で送るメッセージが１つのハッシュ値だけとなり、大幅に通信量を削減することが可能になる。但し、このハッシュ値、及び証明者から送られるチャレンジに対する回答から復元できないハッシュ値については、回答と併せて証明者から送ってもらうようにする。この構成によると、Ｎ回の並列的繰り返し構成の場合、送るべき情報の数を２Ｎ－１個削減することが可能になる。

【０１５２】

（好適なパラメータの設定方法について）

さて、本実施形態に係る対話プロトコルは、受動的攻撃に対する安全性レベルを保証している。しかし、この対話プロトコルを並列的に繰り返し実行する上記の方法を適用した場合、能動的攻撃に対する安全性レベルが確実に保証されているということを保証するには、以下で説明するような条件が必要になる。

【０１５３】

上記の対話プロトコルは、１組の鍵ペア（公開鍵ｙ、秘密鍵ｓ）を用いて「ｙについてｙ＝Ｆ（ｓ）となるｓを知っていること」を証明者が検証者に対して証明する方式であった。そのため、検証で受理される対話を行った場合、「対話の際に証明者がｓを用いた」という情報を検証者に知られてしまう可能性が否定できない。また、上記のＦには衝突困難性が保証されてない。そのため、上記の対話プロトコルを並列的に繰り返し実行する場合について、能動的攻撃に対する安全性レベルが確実に保証されているということを無条件で証明することは難しいのである。

【０１５４】

そこで、本件発明者は、検証で受理される対話を行った場合においても、「対話の際に証明者がｓを用いた」という情報を検証者に知られないようにする方法について検討した。そして、本件発明者は、上記の対話プロトコルを並列的に繰り返し実行する場合においても、能動的攻撃に対する安全性を保証することが可能になる方法を考案した。この方法は、公開鍵として用いるｎ変数多次多項式f_1，…，f_mの数ｍをその変数の数ｎよりも十分に小さな値に設定するというものである。例えば、$2^{m-n} \ll 1$となるようにｍ及びｎが設定される（例えば、ｎ＝１６０、ｍ＝８０の場合$2^{-80} \ll 1$である。）。

【０１５５】

上記のような多次多変数方程式の求解問題に対する解答の困難性に安全性の根拠をおく方式において、秘密鍵s_1と、それに対応する公開鍵ｐｋとが与えられても、その公開鍵ｐｋに対応する別の秘密鍵s_2を生成することは難しい。そのため、公開鍵ｐｋに対する秘密鍵ｓが２つ以上存在することを保証すれば、検証で受理される対話を行った場合においても、「対話の際に証明者がｓを用いた」という情報を検証者に知られないようにすることが可能になる。つまり、この保証が成り立てば、対話プロトコルを並列的に繰り返し実行する場合においても、能動的攻撃に対する安全性を保証することが可能になる。

【０１５６】

図２８を参照しながら、ｍ本のｎ次多変数多項式で構成される関数$F: K^n \to K^m$について考えると（但し、ｎ＞ｍ）、２つ目の原像を持たない定義域の要素数は、最大で$|K|^m - 1$個しか存在しない。そのため、$|K|^{m-n}$を十分に小さくすると、２つ目の原像を持たない定義域の要素が選択される確率を無視できる程度まで小さくすることができる。つまり、ｎ変数多次多項式f_1，…，f_mの数ｍがその変数の数ｎよりも十分に小さな値に設定されていれば、公開鍵ｐｋに対して２つ以上の秘密鍵ｓが存在することを保証できる。その結果、検証で受理される対話を行った場合においても、「対話の際に証明者がｓを用いた」という情報を検証者に知られないようにすることが可能になり、対話プロ

トコルを並列的に繰り返し実行する場合においても、能動的攻撃に対する安全性が保証される。

【０１５７】

以上説明したように、ｎ変数多次多項式f_1，…，f_mの数ｍがその変数の数ｎよりも小さな値（ｎ＞ｍ；好ましくは$2^{m-n} \ll 1$）に設定する方法を適用することで、対話プロトコルを並列的に繰り返し実行する場合の安全性を向上させることが可能になる。

【０１５８】

［２－４：非対話型アルゴリズム］

これまで３パスの公開鍵認証方式について説明してきた。しかし、本手法においては、２パス目で検証者から証明者に送られる情報が検証パターンを示す要求ｄ（実際には単なる乱数を用いることが多い。）だけであるため、１パスの公開鍵認証方式（以下、非対話型方式）に変形することができる。なお、非対話型方式に係る各アルゴリズムの内容を図７に示した。以下、図７を参照しながら、非対話型方式に係る各アルゴリズムの内容について説明する。

【０１５９】

（鍵生成アルゴリズムＧｅｎ）

鍵生成アルゴリズムＧｅｎは、環Ｋ上で定義されるｍ本のｎ変数多次多項式$f_1(x_1, \cdots, x_n), \cdots, f_m(x_1, \cdots, x_n)$と、ベクトル$s=(s_1, \cdots, s_n) \in K^n$を生成する。次に、鍵生成アルゴリズムＧｅｎは、$y=(y_1, \cdots, y_m) \leftarrow (f_1(s), \cdots, f_m(s))$を計算する。そして、鍵生成アルゴリズムＧｅｎは、$(f_1, \cdots, f_m, y)$を公開鍵ｐｋに設定し、ｓを秘密鍵に設定する。なお、以下では、ｎ変数のベクトル$(x_1, \cdots, x_n)$をｘと表記し、ｍ本のｎ変数多次多項式$(f_1(x), \cdots, f_m(x))$をＦ（ｘ）と表記する。

【０１６０】

（証明者アルゴリズムＰ、検証者アルゴリズムＶ）

次に、図７を参照しながら、対話プロトコルにおける証明者アルゴリズムＰと検証者アルゴリズムＶによる処理について説明する。この対話プロトコルは、「証明者がｙ＝Ｆ（ｓ）を満たすｓを知っていること」をｓの情報を検証者に一切漏らさずに、検証者に証明させるものである。なお、鍵生成アルゴリズムＧｅｎにより生成された公開鍵ｐｋは、証明者と検証者の間で共有されているものとする。また、鍵生成アルゴリズムＧｅｎにより生成された秘密鍵ｓは、証明者により秘密に管理されているものとする。

【０１６１】

Ｓｔｅｐ．１：

まず、証明者アルゴリズムＰは、ｉ＝１～Ｎについて、以下の処理（１）～処理（８）を実行する。（処理（１））証明者アルゴリズムＰは、任意に数w_iを選択する。（処理（２））証明者アルゴリズムＰは、擬似乱数生成器G_1に数w_iを適用してベクトル$r_i \in K^n$と数w'_iを生成する。つまり、証明者アルゴリズムＰは、$(r_i, w'_i) \leftarrow G_1(w_i)$を計算する。（処理（３））証明者アルゴリズムＰは、擬似乱数生成器G_2に数w'_iを適用してｎ変数多項式の組$F'_i(x)$を生成する。つまり、証明者アルゴリズムＰは、$F'_i \leftarrow G_2(w'_i)$を計算する。

【０１６２】

Ｓｔｅｐ．１（続き）：

（処理（４））証明者アルゴリズムＰは、$z_i \leftarrow s - r_i$を計算する。この計算は、秘密鍵ｓをベクトルr_iによりマスクする操作に相当する。（処理（５））証明者アルゴリズムＰは、$F''_i(x) \leftarrow F(x+r_i) + F'_i(x)$を計算する。この計算は、ｘについての多項式の組$F(x+r_i)$を多項式の組$F'_i(x)$によりマスクする操作に相当する。

【０１６３】

Ｓｔｅｐ．１（続き）：

（処理（６））証明者アルゴリズムＰは、$F''_i(z_i)$とz_iのハッシュ値$c_{1,i}$

を生成する。つまり、証明者アルゴリズムＰは、$c_{1,i} \leftarrow H_1(F''_i(z_i), z_i)$を計算する。（処理（７））証明者アルゴリズムＰは、数w'_iのハッシュ値$c_{2,i}$を生成する。つまり、証明者アルゴリズムＰは、$c_{2,i} \leftarrow H_2(w'_i)$を計算する。（処理（８））証明者アルゴリズムＰは、多項式の組F''_iのハッシュ値$c_{3,i}$を生成する。つまり、証明者アルゴリズムＰは、$c_{3,i} \leftarrow H_3(F''_i)$を計算する。なお、上記の$H_1(\cdots)$、$H_2(\cdots)$、$H_3(\cdots)$は、ハッシュ関数である。また、ハッシュ値（$c_{1,i}$、$c_{2,i}$、$c_{3,i}$）はメッセージである。

【０１６４】

Ｓｔｅｐ．２：

次に、証明者アルゴリズムＰは、乱数Ｒを選択する。次いで、証明者アルゴリズムＰは、ｉ＝１～Ｎについて、乱数ＲとＳｔｅｐ．１で生成したメッセージ（$c_{1,i}$、$c_{2,i}$、$c_{3,i}$）をハッシュ関数Ｈに適用して$d=(d_1, \cdots, d_N)$を生成する。

【０１６５】

Ｓｔｅｐ．３：

次に、証明者アルゴリズムＰは、生成したd_iに応じて検証者に送り返す情報σ_iを生成する。ここで、証明者アルゴリズムＰは、ｉ＝１～Ｎについて、以下の処理（１）～（３）を実行する。（処理（１））$d_i=0$の場合、証明者アルゴリズムＰは、情報$\sigma_i = w_i$を生成する。（処理（２））$d_i=1$の場合、証明者アルゴリズムＰは、情報$\sigma_i = (w'_i, z_i)$を生成する。（処理（３））$d_i=2$の場合、証明者アルゴリズムＰは、情報$\sigma_i = (F''_i, z_i)$を生成する。上記の処理（１）～（３）の判定及び処理が実行された後、乱数Ｒ、メッセージ（$c_{1,i}$，$c_{2,i}$，$c_{3,i}$）及び情報σ_i（ｉ＝１～Ｎ）は、証明者アルゴリズムＰにより検証者に送られる。

【０１６６】

Ｓｔｅｐ．４：

検証者アルゴリズムＶは、まず、証明者から受け取った乱数Ｒ、メッセージ（$c_{1,i}$，$c_{2,i}$，$c_{3,i}$）及び情報σ_i（ｉ＝１～Ｎ）をハッシュ関数Ｈに適用して$d=(d_1, \cdots, d_N)$を生成する。次いで、検証者アルゴリズムＶは、情報σ_i（ｉ＝１～Ｎ）を利用して以下の検証処理を実行する。なお、以下の処理は、ｉ＝１～Ｎについて実行される。

【０１６７】

$d_i=0$の場合、検証者アルゴリズムＶは、$(r'_i, w''_i) \leftarrow G_1(\sigma_i)$を計算する。さらに、検証者アルゴリズムＶは、$F'''_i \leftarrow G_2(w''_i)$を計算する。そして、検証者アルゴリズムＶは、$c_{2,i}=H_2(w''_i)$の等号が成り立つか否かを検証する。また、検証者アルゴリズムＶは、$c_{3,i}=H_3(F(x+r'_i)+F'''_i(x))$の等号が成り立つか否かを検証する。検証者アルゴリズムＶは、これらの検証が全て成功した場合に認証成功を示す値１を出力し、検証に失敗があった場合に認証失敗を示す値０を出力する。

【０１６８】

$d_i=1$の場合、検証者アルゴリズムＶは、$(w''_i, z'_i) \leftarrow \sigma_i$とする。さらに、検証者アルゴリズムＶは、$F'''_i \leftarrow G_2(w''_i)$を計算する。そして、検証者アルゴリズムＶは、$c_{1,i}=H_1(F'''_i(z'_i), z'_i)$の等号が成り立つか否かを検証する。また、検証者アルゴリズムＶは、$c_2=H_2(w''_i)$の等号が成り立つか否かを検証する。検証者アルゴリズムＶは、これらの検証が全て成功した場合に認証成功を示す値１を出力し、検証に失敗があった場合に認証失敗を示す値０を出力する。

【０１６９】

$d_i=2$の場合、検証者アルゴリズムＶは、$(F''''_i, z'_i) \leftarrow \sigma_i$とする。そして、検証者アルゴリズムＶは、$c_{1,i}=H_1(F''''_i(z'_i)-y, z'_i)$の等号が成り立つか否かを検証する。さらに、検証者アルゴリズムＶは、$c_{3,i}=H_3(F''''_i)$の等号が成り立つか否かを検証する。検証者アルゴリズムＶは、これらの検証が全て成功した場合に認証成功を示す値１を出力し、検証に失敗があった場合に認証失敗を

示す値０を出力する。

【０１７０】

以上、非対話型方式に係る各アルゴリズムの内容について説明した。上記の通り、非対話型方式は、検証者が検証パターンを選択するために乱数ｄを証明者に送る代わりに、証明者アルゴリズムＰがメッセージ（$c_{1,i}$、$c_{2,i}$、$c_{3,i}$）を利用してｄを生成するというものである。理想的なハッシュ関数Ｈを仮定すれば、ハッシュ値ｄがランダムに振る舞うため、証明者に都合の良いハッシュ値ｄが出ることはない。そのため、上記のような変形を行っても、十分な安全性が確保される。なお、このような変形は、拡張方式などに対しても同様に適用することが可能である。

【０１７１】

［２－５：電子署名方式への変形］

ここで、本手法を電子署名方式へと変形する方法について説明する。なお、ここでは簡単のために、上記の非対話型方式を電子署名方式に変形する方法について述べる。上記の非対話型方式における証明者と検証者を電子署名方式における署名者と検証者に対応させると、証明者のみが検証者を納得させられるという点において、電子署名方式のモデルと類似していることが分かるであろう。こうした考えをもとに、非対話型方式に基づく電子署名方式のアルゴリズム構成について詳細に説明する。

【０１７２】

（鍵生成アルゴリズムＧｅｎ）

鍵生成アルゴリズムＧｅｎは、環Ｋ上で定義されるｍ本のｎ変数多次多項式$f_1(x_1, \cdots, x_n), \cdots, f_m(x_1, \cdots, x_n)$と、ベクトル$s=(s_1, \cdots, s_n) \in K^n$を生成する。次に、鍵生成アルゴリズムＧｅｎは、$y=(y_1, \cdots, y_m) \leftarrow (f_1(s), \cdots, f_m(s))$を計算する。そして、鍵生成アルゴリズムＧｅｎは、$(f_1, \cdots, f_m, y)$を公開鍵ｐｋに設定し、ｓを秘密鍵に設定する。なお、以下では、ｎ変数のベクトル$(x_1, \cdots, x_n)$をｘと表記し、ｍ本のｎ変数多次多項式$(f_1(x), \cdots, f_m(x))$をＦ（ｘ）と表記する。

【０１７３】

（署名生成アルゴリズムＳｉｇ）

署名生成アルゴリズムＳｉｇは、ｉ＝１～Ｎについて、以下の（処理１）～（処理１５）を実行する。なお、署名生成アルゴリズムＳｉｇには、署名鍵$sk=(s_1, \cdots, s_N)$と文書Ｍが入力されているものとする。

【０１７４】

（処理１）署名生成アルゴリズムＳｉｇは、任意に数w_iを選択する。（処理２）署名生成アルゴリズムＳｉｇは、擬似乱数生成器G_1に数w_iを適用してベクトル$r_i \in K^n$と数w'_iを生成する。つまり、署名生成アルゴリズムＳｉｇは、$(r_i, w'_i) \leftarrow G_1(w_i)$を計算する。（処理３）署名生成アルゴリズムＳｉｇは、擬似乱数生成器G_2に数w'_iを適用してｍ本のｎ変数多項式$F'_i(x)=(f'_{1,i}(x), \cdots, f'_{m,i}(x))$を生成する。つまり、署名生成アルゴリズムＳｉｇは、$F'_i \leftarrow G2(w'_i)$を計算する。

【０１７５】

（処理４）署名生成アルゴリズムＳｉｇは、$z_i \leftarrow s_i - r_i$を計算する。この計算は、秘密鍵s_iをベクトルr_iによりマスクする操作に相当する。（処理５）署名生成アルゴリズムＳｉｇは、$F''_i(x) \leftarrow F(x+r_i) + F'_i(x)$を計算する。この計算は、ｘについての多項式の組$F(x+r_i)$を多項式の組$F'_i(x)$によりマスクする操作に相当する。

【０１７６】

（処理６）署名生成アルゴリズムＳｉｇは、$F''_i(z_i)$とz_iのハッシュ値$c_{1,i}$を生成する。つまり、署名生成アルゴリズムＳｉｇは、$c_{1,i} \leftarrow H_1(F''_i(z_i), z_i)$を計算する。（処理７）署名生成アルゴリズムＳｉｇは、数w'_iのハッシュ値$c_{2,i}$を生成する。つまり、署名生成アルゴリズムＳｉｇは、$c_{2,i} \leftarrow H_2(w'$

$_{i}$）を計算する。（処理８）署名生成アルゴリズムＳｉｇは、多項式の組Ｆ”$_{i}$のハッシュ値ｃ$_{3,i}$を生成する。つまり、署名生成アルゴリズムＳｉｇは、ｃ$_{3,i}$←Ｈ$_{3}$（Ｆ”$_{i}$）を計算する。なお、上記のＨ$_{1}$（…）、Ｈ$_{2}$（…）、Ｈ$_{3}$（…）は、ハッシュ関数である。

【０１７７】

（処理９）署名生成アルゴリズムＳｉｇは、乱数Ｒを選択する。（処理１０）署名生成アルゴリズムＳｉｇは、ｉ＝１～Ｎについて、文書Ｍ、乱数Ｒ、及びハッシュ値（ｃ$_{1,i}$，ｃ$_{2,i}$，ｃ$_{3,i}$）をハッシュ関数Ｈに適用してｄ＝（ｄ$_{1}$，…，ｄ$_{N}$）を生成する。つまり、署名生成アルゴリズムＳｉｇは、ｄ＝（ｄ$_{1}$，…，ｄ$_{N}$）←Ｈ（Ｒ，Ｍ，ｃ$_{1,1}$，…，ｃ$_{3,N}$）を計算する。（処理１１）署名生成アルゴリズムＳｉｇは、生成したｄ$_{i}$に応じて情報σ$_{i}$を生成する。

【０１７８】

ここで、署名生成アルゴリズムＳｉｇは、ｉ＝１～Ｎについて、以下の（処理１２）～（処理１４）を実行する。（処理１２）ｄ$_{i}$＝０の場合、署名生成アルゴリズムＳｉｇは、情報σ$_{i}$＝ｗ$_{i}$を生成する。（処理１３）ｄ$_{i}$＝１の場合、署名生成アルゴリズムＳｉｇは、情報σ$_{i}$＝（ｗ’$_{i}$，ｚ$_{i}$）を生成する。（処理１４）ｄ$_{i}$＝２の場合、署名生成アルゴリズムＳｉｇは、情報σ$_{i}$＝（Ｆ”$_{i}$，ｚ$_{i}$）を生成する。

【０１７９】

（処理１５）ｉ＝１～Ｎについて、上記（処理１２）～（処理１４）の判定及び処理が実行された後、署名生成アルゴリズムＳｉｇは、乱数Ｒ、メッセージ（ｃ$_{1,i}$、ｃ$_{2,i}$、ｃ$_{3,i}$）及び情報σ$_{i}$（ｉ＝１～Ｎ）を含む電子署名σ＝（Ｒ，ｃ$_{1,i}$、ｃ$_{2,i}$、ｃ$_{3,i}$，σ$_{1}$，…，σ$_{N}$）を出力する。

【０１８０】

（署名検証アルゴリズムＶｅｒ）

署名検証アルゴリズムＶｅｒは、ｉ＝１～Ｎについて、以下の全ての検証を通過すれば、電子署名σを受理し、１つでも検証を通過しなかった場合には電子署名σを拒否する。なお、署名検証アルゴリズムＶｅｒには、電子署名σと文書Ｍが入力されているものとする。まず、署名検証アルゴリズムＶｅｒは、ｄ＝（ｄ$_{1}$，…，ｄ$_{N}$）←Ｈ（Ｒ，Ｍ，ｃ$_{1,1}$，…，ｃ$_{3,N}$）を計算する。次いで、署名検証アルゴリズムＶｅｒは、ｉ＝１～Ｎについて、以下の（検証１）～（検証３）を実行する。

【０１８１】

（検証１）ｄ$_{i}$＝０の場合、署名検証アルゴリズムＶｅｒは、（ｒ’$_{i}$，ｗ”$_{i}$）←Ｇ$_{1}$（σ$_{i}$）を計算する。次いで、署名検証アルゴリズムＶｅｒは、Ｆ’’’$_{i}$←Ｇ$_{2}$（ｗ”$_{i}$）を計算する。そして、署名検証アルゴリズムＶｅｒは、ｃ$_{2,i}$＝Ｈ$_{2}$（ｗ”$_{i}$）の等号が成り立つか否かを検証する。また、署名検証アルゴリズムＶｅｒは、ｃ$_{3,i}$＝Ｈ$_{3}$（Ｆ（ｘ＋ｒ’$_{i}$）＋Ｆ’’’$_{i}$（ｘ））の等号が成り立つか否かを検証する。署名検証アルゴリズムＶｅｒは、これらの検証が全て成功した場合に電子署名σの受理を示す値１を出力し、検証に失敗があった場合に電子署名σの拒否を示す値０を出力する。

【０１８２】

（検証２）ｄ$_{i}$＝１の場合、署名検証アルゴリズムＶｅｒは、（ｗ”$_{i}$，ｚ’$_{i}$）←σ$_{i}$を計算する。次いで、署名検証アルゴリズムＶｅｒは、Ｆ’’’$_{i}$←Ｇ$_{2}$（ｗ”$_{i}$）を計算する。そして、署名検証アルゴリズムＶｅｒは、ｃ$_{1,i}$＝Ｈ$_{1}$（Ｆ’’’$_{i}$（ｚ’$_{i}$），ｚ’$_{i}$）の等号が成り立つか否かを検証する。また、署名検証アルゴリズムＶｅｒは、ｃ$_{2}$＝Ｈ$_{2}$（ｗ”$_{i}$）の等号が成り立つか否かを検証する。署名検証アルゴリズムＶｅｒは、これらの検証が全て成功した場合に電子署名σの受理を示す値１を出力し、検証に失敗があった場合に電子署名σの拒否を示す値０を出力する。

【０１８３】

（検証３）ｄ$_{i}$＝２の場合、署名検証アルゴリズムＶｅｒは、（Ｆ’’’’$_{i}$，ｚ’$_{i}$）←σ$_{i}$を計算する。次いで、署名検証アルゴリズムＶｅｒは、ｃ$_{1,i}$＝Ｈ$_{1}$（Ｆ’’’’$_{i}$（ｚ’$_{i}$）－ｙ，ｚ’$_{i}$）の等号が成り立つか否かを検証する。さらに、署名検証アルゴリ

ズムＶｅｒは、$c_{3,i}=H_3(F''''_i)$の等号が成り立つか否かを検証する。署名検証アルゴリズムＶｅｒは、これらの検証が全て成功した場合に電子署名σの受理を示す値１を出力し、検証に失敗があった場合に電子署名σの拒否を示す値０を出力する。

【０１８４】

以上、本手法に基づく電子署名方式の各アルゴリズム構成について説明した。同様にして、上記の拡張方式に基づく電子署名方式を構築することも可能である。

【０１８５】

［２－６：具体例］

次に、図８を参照しながら、本手法を実施する際に想定される具体的なアルゴリズム構成について説明する。図８は、本手法の具体例を説明するための説明図である。

【０１８６】

（鍵生成アルゴリズムＧｅｎ）

鍵生成アルゴリズムＧｅｎは、環Ｋ上で定義されるｍ本のｎ変数２次多項式$f_1(x_1,\cdots,x_n),\cdots,f_m(x_1,\cdots,x_n)$と、ベクトル$s=(s_1,\cdots,s_n)\in K^n$を生成する。次に、鍵生成アルゴリズムＧｅｎは、$y=(y_1,\cdots,y_m)\leftarrow(f_1(s),\cdots,f_m(s))$を計算する。そして、鍵生成アルゴリズムＧｅｎは、$(f_1,\cdots,f_m,y)$を公開鍵ｐｋに設定し、ｓを秘密鍵に設定する。なお、以下では、ｎ変数のベクトル$(x_1,\cdots,x_n)$をｘと表記し、ｍ本のｎ変数２次多項式$(f_1(x),\cdots,f_m(x))$をＦ（ｘ）と表記する。但し、ｎ変数２次多項式$f_i(x_1,\cdots,x_n)$は、下記の式（１０）のように表現される。

【０１８７】

【数８】

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

【０１８８】

（証明者アルゴリズムＰ、検証者アルゴリズムＶ）

次に、図８を参照しながら、対話プロトコルにおける証明者アルゴリズムＰと検証者アルゴリズムＶによる処理について説明する。この対話プロトコルは、「証明者がｙ＝Ｆ（ｓ）を満たすｓを知っていること」をｓの情報を検証者に一切漏らさずに、検証者に証明させるものである。なお、鍵生成アルゴリズムＧｅｎにより生成された公開鍵ｐｋは、証明者と検証者の間で共有されているものとする。また、鍵生成アルゴリズムＧｅｎにより生成された秘密鍵ｓは、証明者により秘密に管理されているものとする。

【０１８９】

Ｓｔｅｐ．１：

まず、証明者アルゴリズムＰは、任意に数ｗを選択する。次いで、証明者アルゴリズムＰは、擬似乱数生成器G_1に数ｗを適用してベクトル$r\in K^n$と数ｗ’を生成する。つまり、証明者アルゴリズムＰは、$(r,w')\leftarrow G_1(w)$を計算する。次いで、証明者アルゴリズムＰは、擬似乱数生成器G_2に数ｗ’を適用してｍ本の１次多項式の組$f'_1(x_1,\cdots,x_n),\cdots,f'_m(x_1,\cdots,x_n)$を生成する。つまり、証明者アルゴリズムＰは、$(f'_1,\cdots,f'_m)\leftarrow G_2(w')$を計算する。但し、１次多項式$f'_i(x_1,\cdots,x_n)$は、下記の式（１１）のように表現される。また、ｍ本の１次多項式の組$(f'_1(x),\cdots,f'_m(x))$をＦ’（ｘ）と表記する。

【０１９０】

【数９】

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

…（１１）

【０１９１】

Ｓｔｅｐ．１（続き）：

次いで、証明者アルゴリズムＰは、ｚ←ｓ－ｒを計算する。この計算は、秘密鍵ｓをベクトルｒによりマスクする操作に相当する。さらに、証明者アルゴリズムＰは、Ｆ”（ｘ）←Ｆ（ｘ＋ｒ）＋Ｆ’（ｘ）を計算する。この計算は、ｘについての多項式Ｆ（ｘ＋ｒ）を多項式Ｆ’（ｘ）によりマスクする操作に相当する。ここで、Ｆ（ｘ＋ｒ）の中にはｒに関する情報がｘの１次の項にしか現れないため、ｒに関する情報は全てＦ’（ｘ）によりマスクされている点に注意されたい。

【０１９２】

Ｓｔｅｐ．１（続き）：

次いで、証明者アルゴリズムＰは、Ｆ’（ｚ）とｚのハッシュ値c_1を生成する。つまり、証明者アルゴリズムＰは、$c_1 \leftarrow H_1$（Ｆ’（ｚ），ｚ）を計算する。また、証明者アルゴリズムＰは、数ｗ’のハッシュ値c_2を生成する。つまり、証明者アルゴリズムＰは、$c_2 \leftarrow H_2$（ｗ’）を計算する。さらに、証明者アルゴリズムＰは、多項式Ｆ”のハッシュ値c_3を生成する。つまり、証明者アルゴリズムＰは、$c_3 \leftarrow H_3$（Ｆ”）を計算する。なお、上記のH_1（…）、H_2（…）、H_3（…）は、ハッシュ関数である。また、ハッシュ値（c_1、c_2、c_3）はメッセージである。

【０１９３】

Ｓｔｅｐ．１で生成されたメッセージ（c_1、c_2、c_3）は、検証者に送られる。

【０１９４】

Ｓｔｅｐ．２：

検証者アルゴリズムＶは、３つの検証パターンのうち、どの検証パターンを利用するかを選択する。次いで、検証者アルゴリズムＶは、選択した検証パターンを表す要求ｄ∈｛０，１，２｝を証明者に送る。

【０１９５】

Ｓｔｅｐ．３：

証明者アルゴリズムＰは、検証者から受けた要求ｄに応じて検証者に送り返す情報σを生成する。もしｄ＝０の場合、証明者アルゴリズムＰは、情報σ＝ｗを生成する。また、ｄ＝１の場合、証明者アルゴリズムＰは、情報σ＝（ｗ’，ｚ）を生成する。そして、ｄ＝２の場合、証明者アルゴリズムＰは、情報σ＝（Ｆ”，ｚ）を生成する。このようにして生成された情報σは、証明者アルゴリズムＰにより検証者に送られる。

【０１９６】

Ｓｔｅｐ．４：

検証者アルゴリズムＶは、証明者から受け取った情報σを利用して以下の検証処理を実行する。

【０１９７】

ｄ＝０の場合、検証者アルゴリズムＶは、（ｒ’，ｗ”）←G_1（σ）を計算する。さらに、検証者アルゴリズムＶは、ｆ’’’←G_2（ｗ”）を計算する。そして、検証者アルゴリズムＶは、c_2＝H_2（ｗ”）の等号が成り立つか否かを検証する。また、検証者アルゴリズムＶは、c_3＝H_3（Ｆ（ｘ＋ｒ’）＋Ｆ’’’（ｘ））の等号が成り立つか否かを検証する。検証者アルゴリズムＶは、これらの検証が全て成功した場合に認証成功を示す値１を出力し、検証に失敗があった場合に認証失敗を示す値０を出力する。

【０１９８】
ｄ＝１の場合、検証者アルゴリズムＶは、（ｗ”，ｚ’）←σとする。さらに、検証者アルゴリズムＶは、$F'''\leftarrow G_2$（ｗ”）を計算する。そして、検証者アルゴリズムＶは、$c_1=H_1$（F'''（ｚ’），ｚ’）の等号が成り立つか否かを検証する。また、検証者アルゴリズムＶは、$c_2=H_2$（ｗ”）の等号が成り立つか否かを検証する。検証者アルゴリズムＶは、これらの検証が全て成功した場合に認証成功を示す値１を出力し、検証に失敗があった場合に認証失敗を示す値０を出力する。
【０１９９】
ｄ＝２の場合、検証者アルゴリズムＶは、（F'''，ｚ’）←σを計算する。そして、検証者アルゴリズムＶは、$c_1=H_1$（F'''（ｚ’）－ｙ，ｚ’）の等号が成り立つか否かを検証する。さらに、検証者アルゴリズムＶは、$c_3=H_3$（F'''）の等号が成り立つか否かを検証する。検証者アルゴリズムＶは、これらの検証が全て成功した場合に認証成功を示す値１を出力し、検証に失敗があった場合に認証失敗を示す値０を出力する。
【０２００】
以上、本手法を実施する際に想定される具体的なアルゴリズム構成について説明した。
【０２０１】
［２－７：効率的なアルゴリズム］
ｍ本のｎ変数２次多項式の組（f_1（ｘ），…，f_m（ｘ））は、下記の式（１２）のように表現することができる。但し、ｘ＝（x_1，…，x_n）は、ｎ個の変数を表すベクトルである。また、A_1，…，A_mは、ｎ×ｎ行列である。さらに、b_1，…，b_mはｎ×１ベクトルである。そして、ｃは、ｍ×１ベクトルである。
【０２０２】
【数１０】

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

…（１２）

【０２０３】
この表現を用いると、多項式の組Ｆは、下記の式（１３）及び式（１４）のように表現することができる。この表現が成り立つことは、下記の式（１５）から容易に確認することができる。
【０２０４】
【数１１】

$$F(x_1+x_2)=F(x_1)+F(x_2)+F_b(x_1,x_2)-c$$

…（１３）

$$F_b(x_1,x_2)=\begin{pmatrix} x_2^T(A_1^T+A_1)x_1 \\ \vdots \\ x_2^T(A_m^T+A_m)x_1 \end{pmatrix}$$

…（１４）

$$\begin{aligned} f_l(x_1+x_2) &= (x_1+x_2)^T A_l(x_1+x_2)+b_l^T(x_1+x_2)+c_l \\ &= x_1^T A_l x_1 + x_1^T A_l x_2 + x_2^T A_l x_1 + x_2^T A_l x_2 + b_l^T x_1 + b_l^T x_2 + c_l \\ &= f_l(x_1)+f_l(x_2)-c_l+x_1^T A_l x_2 + x_2^T A_l x_1 \\ &= f_l(x_1)+f_l(x_2)-c_l+x_1^T (A_l^T)^T x_2 + x_2^T A_l x_1 \\ &= f_l(x_1)+f_l(x_2)-c_l+(A_l^T x_1)^T x_2 + x_2^T A_l x_1 \\ &= f_l(x_1)+f_l(x_2)-c_l+x_2^T (A_l^T x_1) + x_2^T A_l x_1 \\ &= f_l(x_1)+x_2^T (A_l^T+A_l)x_1 + f_l(x_2)-c_l \end{aligned}$$

…（１５）

【０２０５】

このようにＦ（ｘ$_1$＋ｘ$_2$）をｘ$_1$に依存する部分と、ｘ$_2$に依存する部分と、ｘ$_1$，ｘ$_2$の両方に依存する部分の３つに分けたとき、ｘ$_1$とｘ$_2$の両方に依存する部分Ｆ$_b$（ｘ$_1$，ｘ$_2$）は、ｘ$_1$，ｘ$_2$について双線形写像になる。この性質を利用すると、効率的な方式を実現することが可能になる。

【０２０６】

例えば、本手法において、ｘに関する多項式の組Ｆ（ｘ＋ｒ）をマスクするために利用する多項式の組Ｆ’（ｘ）を、ベクトルｔ∈Ｋ^n、ｅ∈Ｋ^{m}を用いてＦ’（ｘ）＝Ｆ$_b$（ｘ，ｔ）＋ｅとする変形について考えてみたい。この変形を行うと、ｘに関する多項式の組Ｆ（ｘ＋ｒ）、Ｆ’（ｘ）の和は、下記の式（１６）のように表現することができる。ここで、ｔ’＝ｒ＋ｔ、ｅ’＝Ｆ（ｒ）－ｃ＋ｅとおけば、ｘに関する多項式Ｆ”（ｘ）は、ベクトルｔ’∈Ｋ^n、ｅ’∈Ｋ^{m}により表現することができる。これらの理由から、Ｆ’（ｘ）＝Ｆ$_b$（ｘ，ｔ）＋ｅに設定すれば、Ｆ’とＦ”の表現がＫ^{n}上のベクトルとＫ^{m}上のベクトルにより表現することが可能になり、通信に必要なデータサイズを大幅に減らすことができる。

【０２０７】

【数１２】

$$F(x+r)+F'(x)$$
$$=F(x)+F(r)+F_b(x,r)-c+F_b(x,t)+e$$
$$=F(x)+F_b(x,r+t)+F(r)-c+e$$

…（１６）

【０２０８】

また、上記の変形により、Ｆ”（又はＦ’）からｒに関する情報が一切漏れることはない。例えば、ｅ’，ｔ’（又はｅ，ｔ）を与えられても、ｅ，ｔ（又はｅ’，ｔ’）を知らない限り、ｒの情報を一切知ることはできない。従って、上記の変形を施した本手法は、零知識性が担保された公開鍵認証方式である。以下、図９～図１１を参照しながら、上記の変形を施した本手法に係る効率的なアルゴリズムについて、より詳細に説明する。なお、鍵生成アルゴリズムＧｅｎの構成については先に説明した本手法と実質的に同じであるため、詳細な説明を省略する。

【０２０９】

Ｓｔｅｐ．１：

まず、証明者アルゴリズムＰは、任意に数ｗを選択する。次いで、証明者アルゴリズムＰは、擬似乱数生成器G_1に数ｗを適用してベクトル$r \in K^n$と数ｗ’を生成する。つまり、証明者アルゴリズムＰは、（ｒ，ｗ’）←G_1（ｗ）を計算する。次いで、証明者アルゴリズムＰは、擬似乱数生成器G_2に数ｗ’を適用して２つのベクトル$t \in K^n$、$e \in K^m$を生成する。つまり、証明者アルゴリズムＰは、（ｔ，ｅ）←G_2（ｗ’）を計算する。次いで、証明者アルゴリズムＰは、ｚ←ｓ－ｒを計算する。この計算は、秘密鍵ｓをベクトルｒによりマスクする操作に相当する。さらに、証明者アルゴリズムＰは、ｔ’←ｒ＋ｔを計算する。次いで、証明者アルゴリズムＰは、ｅ’←Ｆ（ｒ）－ｃ＋ｅを計算する。

【０２１０】

Ｓｔｅｐ．１（続き）：

次いで、証明者アルゴリズムＰは、上記の式（１６）に基づいてF_b（ｚ，ｔ）を算出し、F_b（ｚ，ｔ）＋ｅとｚのハッシュ値c_1を生成する。つまり、証明者アルゴリズムＰは、c_1←H_1（F_b（ｚ，ｔ）＋ｅ，ｚ）を計算する。また、証明者アルゴリズムＰは、数ｗ’のハッシュ値c_2を生成する。つまり、証明者アルゴリズムＰは、c_2←H_2（ｗ’）を計算する。さらに、証明者アルゴリズムＰは、２つのベクトルｔ’とｅ’のハッシュ値c_3を生成する。つまり、証明者アルゴリズムＰは、c_3←H_3（ｔ’，ｅ’）を計算する。なお、上記のH_1（…）、H_2（…）、H_3（…）は、ハッシュ関数である。また、ハッシュ値（c_1，c_2，c_3）はメッセージである。

【０２１１】

Ｓｔｅｐ．１で生成されたメッセージ（c_1，c_2，c_3）は、検証者に送られる。

【０２１２】

Ｓｔｅｐ．２：

検証者アルゴリズムＶは、３つの検証パターンのうち、どの検証パターンを利用するかを選択する。次いで、検証者アルゴリズムＶは、選択した検証パターンを表す要求$d \in \{0, 1, 2\}$を証明者に送る。

【０２１３】

Ｓｔｅｐ．３：

証明者アルゴリズムＰは、検証者から受けた要求ｄに応じて検証者に送り返す情報σを生成する。もしｄ＝０の場合、証明者アルゴリズムＰは、情報σ＝ｗを生成する。また、

ｄ＝１の場合、証明者アルゴリズムＰは、情報σ＝（ｗ’，ｚ）を生成する。そして、ｄ＝２の場合、証明者アルゴリズムＰは、情報σ＝（ｔ’，ｅ’，ｚ）を生成する。このようにして生成された情報σは、証明者アルゴリズムＰにより検証者に送られる。

【０２１４】

Ｓｔｅｐ．４：

検証者アルゴリズムＶは、証明者から受け取った情報σを利用して以下の検証処理を実行する。

【０２１５】

ｄ＝０の場合、検証者アルゴリズムＶは、（ｒ’，ｗ”）←G_1（σ）を計算する。さらに、検証者アルゴリズムＶは、（ｔ”，ｅ”）←G_2（ｗ”）を計算する。そして、検証者アルゴリズムＶは、$c_2=H_2$（ｗ”）の等号が成り立つか否かを検証する。また、検証者アルゴリズムＶは、$c_3=H_3$（ｒ’＋ｔ”，Ｆ（ｒ’）－ｃ＋ｅ”）の等号が成り立つか否かを検証する。検証者アルゴリズムＶは、これらの検証が全て成功した場合に認証成功を示す値１を出力し、検証に失敗があった場合に認証失敗を示す値０を出力する。

【０２１６】

ｄ＝１の場合、検証者アルゴリズムＶは、（ｗ”，ｚ’）←σとする。さらに、検証者アルゴリズムＶは、（ｔ”，ｅ”）←G_2（ｗ”）を計算する。そして、検証者アルゴリズムＶは、$c_1=H_1$（F_b（ｚ’，ｔ”）＋ｅ”，ｚ’）の等号が成り立つか否かを検証する。また、検証者アルゴリズムＶは、$c_2=H_2$（ｗ”）の等号が成り立つか否かを検証する。検証者アルゴリズムＶは、これらの検証が全て成功した場合に認証成功を示す値１を出力し、検証に失敗があった場合に認証失敗を示す値０を出力する。

【０２１７】

ｄ＝２の場合、検証者アルゴリズムＶは、（ｔ'''，ｅ'''，ｚ’）←σとする。そして、検証者アルゴリズムＶは、$c_1=H_1$（Ｆ（ｚ’）＋F_b（ｚ’，ｔ'''）＋ｅ'''－ｙ，ｚ’）の等号が成り立つか否かを検証する。さらに、検証者アルゴリズムＶは、$c_3=H_3$（ｔ'''，ｅ'''）の等号が成り立つか否かを検証する。検証者アルゴリズムＶは、これらの検証が全て成功した場合に認証成功を示す値１を出力し、検証に失敗があった場合に認証失敗を示す値０を出力する。

【０２１８】

以上、本手法の効率的なアルゴリズムについて説明した。この効率的なアルゴリズムを利用することにより、通信に必要なデータサイズを大幅に削減することができる。また、対話プロトコルの中で、ｘ＋ｒを多項式Ｆに代入し、ｘについて整理しなおす計算（Ｆ（ｘ＋ｒ）の計算）が必要なくなるため、計算効率も向上する。

【０２１９】

（効率的なアルゴリズムの変形例について）

さて、図９に示した本手法の効率的なアルゴリズムは、その計算効率を維持しながら、図１０、図１１に示すようなアルゴリズムに変形することが可能である。

【０２２０】

（変形例Ａ：図１０に示すアルゴリズム）

例えば、図９に示した効率的なアルゴリズムのＳｔｅｐ．１において、ｅ’←Ｆ（ｒ）－ｃ＋ｅという計算を行っているが、この計算を図１０に示すようにｅ’←Ｆ（ｒ）＋ｅという計算に変形してもよい。但し、この変形を行う場合には、Ｓｔｅｐ．４において検証者が行う検証内容の一部を変形する必要がある。具体的には、Ｓｔｅｐ．４においてｄ＝０の場合に検証者が行う検証内容のうち、$c_3=H_3$（ｒ’＋ｔ”，Ｆ（ｒ’）－ｃ＋ｅ”）の検証を$c_3=H_3$（ｒ’＋ｔ”，Ｆ（ｒ’）＋ｅ”）の検証に置き換える必要がある。また、Ｓｔｅｐ．４においてｄ＝２の場合に検証者が行う検証内容のうち、$c_1=H_1$（Ｆ（ｚ’）＋F_b（ｚ’，ｔ’’’）＋ｅ’’’－ｙ，ｚ’）の検証を$c_1=H_1$（Ｆ（ｚ’）＋F_b（ｚ’，ｔ’’’）－ｃ＋ｅ’’’－ｙ，ｚ’）の検証に置き換える必要がある。

【０２２１】
（変形例Ｂ：図１１に示すアルゴリズム）
また、図１０に示したアルゴリズムのＳｔｅｐ．３において、ｄ＝０の場合にσをｗに設定しているが、ｄ＝０の場合に設定されるσは、（ｒ，ｔ，ｅ）が復元できる情報であればどのような情報でもよい。例えば、図１１に示すように、Ｓｔｅｐ．３においてｄ＝０の場合に設定されるσの内容を（ｗ’，ｔ’）にしてもよい。但し、この変形を行う場合には、Ｓｔｅｐ．４において検証者が行う検証内容の一部を変形する必要がある。具体的には、Ｓｔｅｐ．４においてｄ＝０の場合に検証者が行う検証内容のうち、$c_3=H_3$（ｒ’＋ｔ”，Ｆ（ｒ’）＋ｅ”）の検証を$c_3=H_3$（ｔ’，Ｆ（ｔ’－ｔ）＋ｅ”）の検証に置き換える必要がある。

【０２２２】
上記のように、Ｓｔｅｐ．１において証明者により行われるｅ’←Ｆ（ｒ）－ｃ＋ｅという計算をｅ’←Ｆ（ｒ）＋ｅという計算に変更することができる（変形例Ａ）。また、Ｓｔｅｐ．３においてｄ＝０の場合に証明者により設定されるσの内容を（ｗ’，ｔ’）などに変更することができる（変形例Ｂ）。図１１には、変形例Ａに係る変形と変形例Ｂに係る変形を共に図９の効率的なアルゴリズムに対して適用したアルゴリズムを示したが、変形例Ｂに係る変形だけを図９の効率的なアルゴリズムに対して適用することも可能である。つまり、変形例Ａの変形内容及び変形例Ｂの変形内容は、それぞれ単独で図９のアルゴリズムに適用してもよいし、両者を組み合わせて図９のアルゴリズムに適用してもよい。

【０２２３】
［２－８：多次多変数連立方程式の形式］
これまで説明してきたように、本手法は、多次多変数連立方程式の求解問題に対する解答の困難性に安全性の根拠をおく方式である。また、本手法は、複雑な多次多変数連立方程式を利用可能な点に特徴がある。ここまでの説明においては多次多変数連立方程式の形式に特別な限定を行ってこなかったが、例えば、十分に困難性が補償されている暗号要素技術を表現した多次多変数連立方程式を利用することが好ましい。以下、本手法に適用可能な多次多変数連立方程式の具体例を紹介する。

【０２２４】
（２－８－１：共通鍵ブロック暗号に関する形式）
ＡＥＳ、ＤＥＳ、ＫＡＴＡＮなどの共通鍵ブロック暗号技術は、よく解析され、安全性及び信頼性が高い要素技術である。これらの共通鍵ブロック暗号は、共通鍵ブロック暗号の鍵、平文、暗号文を変数とする多次多変数連立方程式により表現することができる。この多次多変数連立方程式において平文と暗号文を表現する変数に値を与えると、この多次多変数連立方程式は、鍵を表現する変数だけを変数に持つ方程式になる。

【０２２５】
このような共通鍵ブロック暗号を表現した多次多変数連立方程式の解を求めることは、平文と暗号文から共通鍵ブロック暗号の鍵を復元することに相当する。つまり、その共通鍵ブロック暗号の安全性が維持されている限りにおいて、その共通鍵ブロック暗号を表現した多次多変数連立方程式の求解困難性は担保される。そのため、ある共通鍵ブロック暗号方式を表現する多次多変数連立方程式を本手法に適用するならば、共通鍵ブロック暗号方式の安全性と同等の安全性を具備した公開鍵認証方式が実現される。

【０２２６】
但し、共通鍵ブロック暗号を、鍵、平文、暗号文を変数とする多次多変数連立方程式で表現した場合、多項式の次数が大きくなるため、連立方程式を表現するためのデータサイズが増大してしまう。そこで、鍵、平文、暗号文に加え、それぞれのラウンドにおける内部状態を表現するための変数を導入する。この変数を導入すれば、共通鍵ブロック暗号を表現する多次多変数連立方程式の次数を小さくすることができる。例えば、平文と暗号文を表現する変数に適当な値を代入し、鍵と内部状態を表現するための変数に関する連立方程式を導入する。このような方法を採用することにより、変数の数は増加するが、次数が

下がるため、連立方程式の表現はコンパクトになる。
【０２２７】
（２－８－２：ハッシュ関数に関する形式）
同様に、ＳＨＡ－１、ＳＨＡ－２５６などのハッシュ関数に関する多次多変数連立方程式を本手法に適用することも可能である。これらのハッシュ関数は、ハッシュ関数の入力となるメッセージと出力となるハッシュ値を変数とする多次多変数連立方程式で表現することができる。この多次多変数連立方程式において、ハッシュ値を表現する変数に適当な値を与えると、対応する入力を表現した変数に関する多次多変数連立方程式が得られる。
【０２２８】
このような多次多変数連立方程式の解を求めることは、ハッシュ値から、元になるメッセージの値を復元することに相当する。つまり、そのハッシュ関数の安全性（一方向性）が維持されている限りにおいて、そのハッシュ関数を表現した多次多変数連立方程式の求解問題に対する解答の困難性は担保される。そのため、あるハッシュ関数を表現した多次多変数連立方程式を本手法に適用すれば、ハッシュ関数の安全性に基づく公開鍵認証方式が実現される。
【０２２９】
但し、ハッシュ関数を、入力メッセージとハッシュ値を変数とする多次多変数連立方程式で表現した場合、多項式の次数が大きくなるため、連立方程式を表現するためのデータサイズが増大してしまう。そこで、入力メッセージ、ハッシュ値に加え、内部状態を表現するための変数を導入する。この変数を導入すれば、ハッシュ関数を表現する多次多変数連立方程式の次数を小さくすることができる。例えば、ハッシュ値を表現する変数に適用な値を代入し、入力メッセージと内部状態を表現するための変数に関する連立方程式を導入する。このような方法を採用することにより、変数の数は増加するが、次数が下がるため、連立方程式の表現はコンパクトになる。
【０２３０】
（２－８－３：ストリーム暗号に関する形式）
同様に、Ｔｒｉｖｉｕｍなどのストリーム暗号に関する多次多変数連立方程式を本手法に適用することも可能である。これらのストリーム暗号は、ストリーム暗号の初期の内部状態を表現する変数と、出力されるストリームを表現する変数に関する多次多変数連立方程式で表現することが可能である。この場合、出力ストリームを表現する変数に適当な値を与えると、対応する初期の内部状態を表現するための変数に関する多次多変数連立方程式が得られる。
【０２３１】
このような多次多変数連立方程式の解を求めることは、出力されたストリームの値から、元になった初期の内部状態を表現する変数を復元することに相当する。つまり、そのストリーム暗号の安全性が確保されている限りにおいて、そのストリーム暗号を表現した多次多変数連立方程式の求解問題に対する解答の困難性は担保されている。そのため、あるストリーム暗号を表現した多次多変数連立方程式を本手法に適用すれば、ストリーム暗号の安全性に基づく公開鍵認証方式が実現される。
【０２３２】
但し、ストリーム暗号を、初期の内部状態と出力ストリームを変数とする多次多変数連立方程式で表現した場合、多項式の次数が大きくなり、連立方程式を表現するサイズが増大してしまう。そこで、初期の内部状態と出力ストリームに加え、それぞれのラウンドにおける内部状態を表現するための変数を導入する。この変数を導入すれば、ストリーム暗号を表現するための多次多変数連立方程式の次数を小さくすることが可能になる。例えば、出力ストリームを表現する変数に適当な値を代入し、初期の内部状態とラウンドにおける内部状態を表現するための変数に関する連立方程式を導入する。このような方法を採用することにより、変数の数は増加するが、次数が下がるため、連立方程式の表現はコンパクトになる。
【０２３３】

以上、本発明の第１実施形態について説明した。

【０２３４】

＜３：第２実施形態＞

次に、本発明の第２実施形態について説明する。これまで３パスの公開鍵認証方式について説明してきた。本実施形態では、５パスの公開鍵認証方式（以下、本手法）について説明する。本手法は、検証者の検証パターンを２ｑ通りにすることにより、公開鍵認証方式の健全性を確保する方式である。

【０２３５】

なお、上記の第１実施形態に係る３パスの公開鍵認証方式は、対話プロトコル１回当たりの偽証確率が２／３であったが、本手法では、後述するように対話プロトコル１回当たりの偽証確率は１／２＋１／ｑとなる。但し、ｑは、利用する環の位数である。従って、環の位数が十分に大きい場合、図２７に示すように、本手法の方が１回当たりの偽証確率を低減することが可能になり、少ない対話プロトコルの実行回数で、偽証確率を十分に小さくすることができる。

【０２３６】

５パスの公開鍵認証方式に係る対話プロトコルは、３パスの公開鍵認証方式に係る対話プロトコルよりも効率が低いように思われるかもしれない。しかし、５パスの公開鍵認証方式においては、環の位数を十分に大きくとった場合、対話プロトコル１回当たりの偽証確率が１／２に近くなるため、同じセキュリティレベルを達成するために必要な対話プロトコルの実行回数が少なくて済む。

【０２３７】

例えば、偽証確率を１／２^{n}以下にしたい場合、３パスの公開鍵認証方式においては、ｎ／（ｌｏｇ３－１）＝１．７０１ｎ回以上、対話プロトコルを実行する必要がある。一方、５パスの公開鍵認証方式においては、ｎ／（１－ｌｏｇ（１＋１／ｑ））回以上、対話プロトコルを実行する必要がある。図２７に示すように、例えば、ｑ＝２４にすれば、同じセキュリティレベルを実現するのに必要な通信量は、３パスの公開鍵認証方式に比べ、５パスの公開鍵認証方式の方が少なくなる。

【０２３８】

［３－１：公開鍵認証方式のアルゴリズム］

以下、図１２を参照しながら、５パスの公開鍵認証方式（本手法）に係るアルゴリズム構成について説明する。図１２は、本手法に係るアルゴリズムの内容について説明するための説明図である。

【０２３９】

（鍵生成アルゴリズムＧｅｎ）

鍵生成アルゴリズムＧｅｎは、環Ｋ上で定義されるｍ本のｎ変数多次多項式$f_1(x_1,\cdots,x_n),\cdots,f_m(x_1,\cdots,x_n)$と、ベクトル$s=(s_1,\cdots,s_n)\in K^n$を生成する。次に、鍵生成アルゴリズムＧｅｎは、$y=(y_1,\cdots,y_m)\leftarrow f_1(s),\cdots,f_m(s)$を計算する。そして、鍵生成アルゴリズムＧｅｎは、$(f_1,\cdots,f_m,y)$を公開鍵ｐｋに設定し、ｓを秘密鍵に設定する。なお、以下では、ｎ変数のベクトル$(x_1,\cdots,x_n)$をｘと表記し、ｍ本のｎ変数多次多項式$(f_1(x),\cdots,f_m(x))$をＦ（ｘ）と表記する。

【０２４０】

（証明者アルゴリズムＰ、検証者アルゴリズムＶ）

次に、図１２を参照しながら、対話プロトコルにおける証明者アルゴリズムＰと検証者アルゴリズムＶによる処理について説明する。この対話プロトコルは、「証明者がｙ＝Ｆ（ｓ）を満たすｓを知っていること」をｓの情報を検証者に一切漏らさずに、検証者に証明させるものである。なお、鍵生成アルゴリズムＧｅｎにより生成された公開鍵ｐｋは、証明者と検証者の間で共有されているものとする。また、鍵生成アルゴリズムＧｅｎにより生成された秘密鍵ｓは、証明者により秘密に管理されているものとする。

【０２４１】

Ｓｔｅｐ．１：
まず、証明者アルゴリズムＰは、任意に数ｗを選択する。次いで、証明者アルゴリズムＰは、擬似乱数生成器Ｇに数ｗを適用してベクトル$r \in K^n$と、ｎ変数多項式の組$F'(x)=(f'_1(x),\cdots,f'_m(x))$を生成する。つまり、証明者アルゴリズムＰは、（ｒ，Ｆ’）←Ｇ（ｗ）を計算する。次いで、証明者アルゴリズムＰは、ｚ←ｓ－ｒを計算する。この計算は、秘密鍵ｓをベクトルｒによりマスクする操作に相当する。

【０２４２】
Ｓｔｅｐ．１（続き）：
次いで、証明者アルゴリズムＰは、Ｆ’（ｚ）とｚのハッシュ値c_1を生成する。つまり、証明者アルゴリズムＰは、$c_1 \leftarrow H_1(F'(z), z)$を計算する。また、証明者アルゴリズムＰは、数ｗのハッシュ値c_2を生成する。つまり、証明者アルゴリズムＰは、$c_2 \leftarrow H_2(w)$を計算する。なお、上記の$H_1(\cdots)$、$H_2(\cdots)$は、ハッシュ関数である。また、ハッシュ値(c_1, c_2)はメッセージである。

【０２４３】
Ｓｔｅｐ．１で生成されたメッセージ(c_1, c_2)は、検証者に送られる。

【０２４４】
Ｓｔｅｐ．２：
検証者アルゴリズムＶは、ｑ通り存在する環Ｋの元から１つの乱数αを選択する。そして、検証者アルゴリズムＶは、選択した乱数αを証明者に送る。

【０２４５】
Ｓｔｅｐ．３：
証明者アルゴリズムＰは、$F''(x) \leftarrow \alpha F(x+r) + F'(x)$を計算する。この計算は、ｘについての多項式の組$F(x+r)$を多項式の組$F'(x)$によりマスクする操作に相当する。

【０２４６】
Ｓｔｅｐ．３で生成された多項式Ｆ”は、検証者に送られる。

【０２４７】
Ｓｔｅｐ．４：
検証者アルゴリズムＶは、２つの検証パターンのうち、どちらの検証パターンを利用するかを選択する。次いで、検証者アルゴリズムＶは、選択した検証パターンを表す要求$d \in \{0, 1\}$を証明者に送る。

【０２４８】
Ｓｔｅｐ．５：
証明者アルゴリズムＰは、検証者から受けた要求ｄに応じて検証者に送り返す情報σを生成する。もしｄ＝０の場合、証明者アルゴリズムＰは、情報σ＝ｗを生成する。また、ｄ＝１の場合、証明者アルゴリズムＰは、情報σ＝ｚを生成する。このようにして生成された情報σは、証明者アルゴリズムＰにより検証者に送られる。

【０２４９】
Ｓｔｅｐ．６：
検証者アルゴリズムＶは、証明者から受け取った情報σを利用して以下の検証処理を実行する。

【０２５０】
ｄ＝０の場合、検証者アルゴリズムＶは、$(r', F''') \leftarrow G(\sigma)$を計算する。そして、検証者アルゴリズムＶは、$c_2 = H_2(\sigma)$の等号が成り立つか否かを検証する。また、検証者アルゴリズムＶは、$F''(x) = \alpha F(x+r') + F'''(x)$の等号が成り立つか否かを検証する。検証者アルゴリズムＶは、これらの検証が全て成功した場合に認証成功を示す値１を出力し、検証に失敗があった場合に認証失敗を示す値０を出力する。

【０２５１】
ｄ＝１の場合、検証者アルゴリズムＶは、$z' \leftarrow \sigma$を計算する。そして、検証者アルゴ

リズムＶは、$c_1 = H_1(F'''(z') - \alpha y, z')$の等号が成り立つか否かを検証する。検証者アルゴリズムＶは、この検証が成功した場合に認証成功を示す値１を出力し、検証に失敗があった場合に認証失敗を示す値０を出力する。

【０２５２】

（補足）

さて、上記Ｓｔｅｐ．１及びＳｔｅｐ．３において生成されたメッセージ（c_1，c_2）及び多項式の組Ｆ”を検証者に送った際、秘密鍵ｓｋに関する情報、ｒに関する情報、ｚに関する情報が検証者に一切漏れていないことに注意されたい。また、上記Ｓｔｅｐ．５において生成された情報σを検証者に送った際、ｄ＝０のときにはｚに関する情報が検証者に一切漏れておらず、ｄ＝１のときにはｒに関する情報が検証者に一切漏れていないことに注意されたい。

【０２５３】

（本手法における健全性について）

本手法の健全性は、証明者アルゴリズムＰが１組の（c_1，c_2）、及び、検証者アルゴリズムＶが選択する２通りの（α_1，α_2）について、要求ｄ＝０，１に正しく回答した場合に、その回答内容から下記の式（１７）～式（１９）を満たすF''''_1、F''''_2、F'''、ｒ’、ｚ’を計算できるということから保証されている。

【０２５４】

【数１３】

$$F''''_1(x) = \alpha_1 F(x + r') + F'''(x)$$

…（１７）

$$F''''_2(x) = \alpha_2 F(x + r') + F'''(x)$$

…（１８）

$$F''''_1(z') - \alpha_1 y = F''''_2(z') - \alpha_2 y$$

…（１９）

【０２５５】

このようなロジックが保証されることにより、多変数連立方程式の求解問題を解かない限りにおいて１／２＋１／ｑより高い確率で偽証することは不可能であることが保証される。つまり、検証者の要求ｄ＝０，１の全てに正しく回答するためには、偽証者が上記の式（１７）～式（１９）を満たすF''''_1、F''''_2、F'''、ｒ’、ｚ’を計算できる必要がある。言い換えると、偽証者はＦ（ｓ）＝ｙを満たすｓを計算できる必要がある。従って、多変数連立方程式の求解問題を解かない限り、偽証者は、１／２＋１／ｑより高い確率で偽証を成功させることができないのである。なお、上記の対話プロトコルを十分に多くの回数実行することにより、偽証の成功確率を無視できるほど小さくすることができる。

【０２５６】

（変形例）

上記の鍵生成アルゴリズムＧｅｎは、ｙ←Ｆ（ｓ）を計算し、公開鍵を（Ｆ，ｙ）に設定している。しかし、鍵生成アルゴリズムＧｅｎは、$(y_1, \cdots, y_m) \leftarrow F(s)$として、$(f_1^*(x), \cdots, f_m^*(x)) \leftarrow (f_1(x) - y_1, \cdots, f_m(x) - y_m)$を計算し、公開鍵を$(f_1^*, \cdots, f_m^*)$に設定するように構成されていてもよい。このように変形すると、証明者と検証者との間でｙ＝０として対話プロトコルを実行することが可能になる。

【０２５７】

また、証明者アルゴリズムＰは、Ｆ”（ｚ）のハッシュ値と、ｚのハッシュ値とを別々に計算し、それぞれをメッセージとして検証者に送るようにしてもよい。

【０２５８】

また、上記の証明者アルゴリズムＰは、数ｗに擬似乱数生成器$Ｇ_{1}$を適用してベクトルｒと数ｗ’を生成している。そして、上記の証明者アルゴリズムＰは、数ｗ’を擬似乱数生成器$Ｇ_{2}$に適用してｎ変数多項式Ｆ’（ｘ）を生成している。しかし、証明者アルゴリズムＰは、最初からｗ＝（ｒ，Ｆ’）を算出し、$Ｇ_{1}$を恒等写像としてもよい。また、この場合には、数ｗを$Ｇ_{1}$に適用する必要はない。なお、$Ｇ_{2}$についても同様である。

【０２５９】

以上、本手法に係る基本的なアルゴリズム構成について説明した。

【０２６０】

［３－２：拡張アルゴリズム］

次に、図１３を参照しながら、本手法を拡張した公開鍵認証方式（以下、拡張手法）のアルゴリズムについて説明する。図１３は、拡張手法に基づく対話プロトコルの流れを説明するための説明図である。この拡張手法は、３パス目に送信する多項式の組Ｆ”を１つのハッシュ値$ｃ_{3}$に変換して検証者に送る方式である。このように拡張することで、対話プロトコルの中で表現サイズの大きい多項式の組Ｆ”を検証者に送る確率を半減することが可能になり、通信する平均的なデータサイズを削減することができる。以下、拡張方式における各アルゴリズムの内容について詳細に説明する。

【０２６１】

（鍵生成アルゴリズムＧｅｎ）

鍵生成アルゴリズムＧｅｎは、環Ｋ上で定義されるｍ本のｎ変数多次多項式$ｆ_{1}（ｘ_{1}，…，ｘ_{n}），…，ｆ_{m}（ｘ_{1}，…，ｘ_{n}）$と、ベクトル$ｓ＝（ｓ_{1}，…，ｓ_{n}）∈Ｋ^{n}$を生成する。次に、鍵生成アルゴリズムＧｅｎは、$ｙ＝（ｙ_{1}，…，ｙ_{m}）←（ｆ_{1}（ｓ），…，ｆ_{m}（ｓ））$を計算する。そして、鍵生成アルゴリズムＧｅｎは、$（ｆ_{1}，…，ｆ_{m}，ｙ）$を公開鍵ｐｋに設定し、ｓを秘密鍵に設定する。なお、以下では、ｎ変数のベクトル$（ｘ_{1}，…，ｘ_{n}）$をｘと表記し、ｍ本のｎ変数多次多項式$（ｆ_{1}（ｘ），…，ｆ_{m}（ｘ））$をＦ（ｘ）と表記する。

【０２６２】

（証明者アルゴリズムＰ、検証者アルゴリズムＶ）

次に、図１３を参照しながら、対話プロトコルにおける証明者アルゴリズムＰと検証者アルゴリズムＶによる処理について説明する。この対話プロトコルは、「証明者がｙ＝Ｆ（ｓ）を満たすｓを知っていること」をｓの情報を検証者に一切漏らさずに、検証者に証明させるものである。なお、鍵生成アルゴリズムＧｅｎにより生成された公開鍵ｐｋは、証明者と検証者の間で共有されているものとする。また、鍵生成アルゴリズムＧｅｎにより生成された秘密鍵ｓは、証明者により秘密に管理されているものとする。

【０２６３】

Ｓｔｅｐ．１：

まず、証明者アルゴリズムＰは、任意に数ｗを選択する。次いで、証明者アルゴリズムＰは、擬似乱数生成器Ｇに数ｗを適用してベクトル$ｒ∈Ｋ^{n}$と、ｎ変数多項式の組Ｆ’（ｘ）を生成する。つまり、証明者アルゴリズムＰは、（ｒ，Ｆ’）←Ｇ（ｗ）を計算する。次いで、証明者アルゴリズムＰは、ｚ←ｓ－ｒを計算する。この計算は、秘密鍵ｓをベクトルｒによりマスクする操作に相当する。

【０２６４】

Ｓｔｅｐ．１（続き）：

次いで、証明者アルゴリズムＰは、Ｆ’（ｚ）とｚのハッシュ値$ｃ_{1}$を生成する。つまり、証明者アルゴリズムＰは、$ｃ_{1}←Ｈ_{1}（Ｆ’（ｚ），ｚ）$を計算する。また、証明者アルゴリズムＰは、数ｗのハッシュ値$ｃ_{2}$を生成する。つまり、証明者アルゴリズムＰは、$ｃ_{2}←Ｈ_{2}（ｗ）$を計算する。なお、上記の$Ｈ_{1}（…）$、$Ｈ_{2}（…）$は、ハッシュ関数である。また、ハッシュ値$（ｃ_{1}，ｃ_{2}）$はメッセージである。

【０２６５】
Ｓｔｅｐ．１で生成されたメッセージ（c_1，c_2）は、検証者に送られる。
【０２６６】
Ｓｔｅｐ．２：
検証者アルゴリズムＶは、ｑ通り存在する環Ｋの元から１つの乱数αを選択する。そして、検証者アルゴリズムＶは、選択した乱数αを証明者に送る。
【０２６７】
Ｓｔｅｐ．３：
証明者アルゴリズムＰは、Ｆ”（ｘ）←αＦ（ｘ＋ｒ）＋Ｆ’（ｘ）を計算する。この計算は、ｘについての多項式の組Ｆ（ｘ＋ｒ）を多項式の組Ｆ’（ｘ）によりマスクする操作に相当する。さらに、証明者アルゴリズムＰは、多項式の組Ｆ”のハッシュ値c_3を生成する。つまり、証明者アルゴリズムＰは、$c_3 \leftarrow H_3$（Ｆ”（ｘ））を計算する。なお、上記のH_3（…）は、ハッシュ関数である。また、ハッシュ値c_3はメッセージである。
【０２６８】
Ｓｔｅｐ．３で生成されたメッセージc_3は、検証者に送られる。
【０２６９】
Ｓｔｅｐ．４：
検証者アルゴリズムＶは、２つの検証パターンのうち、どちらの検証パターンを利用するかを選択する。次いで、検証者アルゴリズムＶは、選択した検証パターンを表す要求ｄ∈｛０，１｝を証明者に送る。
【０２７０】
Ｓｔｅｐ．５：
証明者アルゴリズムＰは、検証者から受けた要求ｄに応じて検証者に送り返す情報σを生成する。もしｄ＝０の場合、証明者アルゴリズムＰは、情報σ＝ｗを生成する。また、ｄ＝１の場合、証明者アルゴリズムＰは、情報σ＝（ｚ，Ｆ”）を生成する。このようにして生成された情報σは、証明者アルゴリズムＰにより検証者に送られる。
【０２７１】
Ｓｔｅｐ．６：
検証者アルゴリズムＶは、証明者から受け取った情報σを利用して以下の検証処理を実行する。
【０２７２】
ｄ＝０の場合、検証者アルゴリズムＶは、（ｒ’，Ｆ’’’）←Ｇ（σ）を計算する。そして、検証者アルゴリズムＶは、$c_2 = H_2$（σ）の等号が成り立つか否かを検証する。また、検証者アルゴリズムＶは、$c_3 = H_3$（αＦ（ｘ＋ｒ’）＋Ｆ’’’（ｘ））の等号が成り立つか否かを検証する。検証者アルゴリズムＶは、これらの検証が全て成功した場合に認証成功を示す値１を出力し、検証に失敗があった場合に認証失敗を示す値０を出力する。
【０２７３】
ｄ＝１の場合、検証者アルゴリズムＶは、（ｚ’，Ｆ’’’）←σを計算する。そして、検証者アルゴリズムＶは、$c_1 = H_1$（Ｆ’’’（ｚ’）－αｙ，ｚ’）の等号が成り立つか否かを検証する。また、検証者アルゴリズムＶは、$c_2 = H_2$（Ｆ’’’（ｘ））の等号が成り立つか否かを検証する。検証者アルゴリズムＶは、これらの検証が全て成功した場合に認証成功を示す値１を出力し、検証に失敗があった場合に認証失敗を示す値０を出力する。
【０２７４】
以上、拡張方式の対話プロトコルにおける各アルゴリズムの処理について説明した。このように拡張することで、対話プロトコルの中で表現サイズの大きい多項式の組Ｆ”を検証者に送る確率を半減することが可能になり、通信する平均的なデータサイズを削減することができる。

【０２７５】
［３－３：並列化アルゴリズム］
さて、先に述べた通り、本手法又は拡張手法に係る対話プロトコルを適用すれば、偽証が成功する確率を（１／２＋１／ｑ）以下に抑制することができる。従って、この対話プロトコルを２回実行すれば、偽証が成功する確率を（１／２＋１／ｑ）^{2}以下に抑制することができる。同様に、この対話プロトコルをＮ回実行すると、偽証が成功する確率は（１／２＋１／ｑ）^{N}となり、Ｎを十分に大きい数（例えば、Ｎ＝８０）にすれば、偽証が成功する確率は無視できる程度に小さくなる。例えば、本手法に係る対話プロトコルを並列的にＮ回実行するアルゴリズムは図１４のようになる。以下、図１４を参照しながら、並列的に対話プロトコルをＮ回実行する各アルゴリズムの内容について説明する。
【０２７６】
（鍵生成アルゴリズムＧｅｎ）
鍵生成アルゴリズムＧｅｎは、環Ｋ上で定義されるｍ本のｎ変数多次多項式$ｆ_{1}$（$ｘ_{1}$，…，$ｘ_{ｎ}$），…，$ｆ_{ｍ}$（$ｘ_{1}$，…，$ｘ_{ｎ}$）と、ベクトル$ｓ＝（ｓ_{1}，…，ｓ_{ｎ}）∈Ｋ^{ｎ}$を生成する。次に、鍵生成アルゴリズムＧｅｎは、$ｙ＝（ｙ_{1}，…，ｙ_{ｍ}）←（ｆ_{1}$（ｓ），…，$ｆ_{ｍ}$（ｓ））を計算する。そして、鍵生成アルゴリズムＧｅｎは、（$ｆ_{1}$，…，$ｆ_{ｍ}$，ｙ）を公開鍵ｐｋに設定し、ｓを秘密鍵に設定する。なお、以下では、ｎ変数のベクトル（$ｘ_{1}$，…，$ｘ_{ｎ}$）をｘと表記し、ｍ本のｎ変数多次多項式（$ｆ_{1}$（ｘ），…，$ｆ_{ｍ}$（ｘ））をＦ（ｘ）と表記する。
【０２７７】
（証明者アルゴリズムＰ、検証者アルゴリズムＶ）
次に、図１４を参照しながら、対話プロトコルにおける証明者アルゴリズムＰと検証者アルゴリズムＶによる処理について説明する。この対話プロトコルは、「証明者がｙ＝Ｆ（ｓ）を満たすｓを知っていること」をｓの情報を検証者に一切漏らさずに、検証者に証明させるものである。なお、鍵生成アルゴリズムＧｅｎにより生成された公開鍵ｐｋは、証明者と検証者の間で共有されているものとする。また、鍵生成アルゴリズムＧｅｎにより生成された秘密鍵ｓは、証明者により秘密に管理されているものとする。
【０２７８】
Ｓｔｅｐ．１：
まず、証明者アルゴリズムＰは、ｉ＝１～Ｎについて、以下の処理（１）～処理（５）を実行する。（処理（１））証明者アルゴリズムＰは、任意に数$ｗ_{ｉ}$を選択する。（処理（２））証明者アルゴリズムＰは、擬似乱数生成器Ｇに数$ｗ_{ｉ}$を適用してベクトル$ｒ_{ｉ}∈Ｋ^{ｎ}$と多項式の組$Ｆ’_{ｉ}$（ｘ）を生成する。つまり、証明者アルゴリズムＰは、（$ｒ_{ｉ}$，$Ｆ’_{ｉ}$）←Ｇ（$ｗ_{ｉ}$）を計算する。（処理（３））証明者アルゴリズムＰは、$ｚ_{ｉ}←ｓ－ｒ_{ｉ}$を計算する。この計算は、秘密鍵ｓをベクトル$ｒ_{ｉ}$によりマスクする操作に相当する。
【０２７９】
Ｓｔｅｐ．１（続き）：
（処理（４））証明者アルゴリズムＰは、$Ｆ’_{ｉ}$（$ｚ_{ｉ}$）と$ｚ_{ｉ}$のハッシュ値$ｃ_{1，ｉ}$を生成する。つまり、証明者アルゴリズムＰは、$ｃ_{1，ｉ}←Ｈ_{1}$（$Ｆ’_{ｉ}$（$ｚ_{ｉ}$），$ｚ_{ｉ}$）を計算する。（処理（５））証明者アルゴリズムＰは、数$ｗ’_{ｉ}$のハッシュ値$ｃ_{2，ｉ}$を生成する。つまり、証明者アルゴリズムＰは、$ｃ_{2，ｉ}←Ｈ_{2}$（$ｗ’_{ｉ}$）を計算する。
【０２８０】
ｉ＝１～Ｎについて、上記の処理（１）～（５）が実行された後、Ｓｔｅｐ．１で生成されたメッセージ（$ｃ_{1，ｉ}$，$ｃ_{2，ｉ}$）（ｉ＝１～Ｎ）は、検証者に送られる。
【０２８１】
Ｓｔｅｐ．２：
検証者アルゴリズムＶは、ｑ通り存在する環Ｋの元からＮ個の乱数$α_{1}$，…，$α_{Ｎ}$を選択する。そして、検証者アルゴリズムＶは、選択した乱数$α_{1}$，…，$α_{Ｎ}$を証明者に送る。

【０２８２】
Ｓｔｅｐ．３：
検証者アルゴリズムＶは、ｉ＝１～Ｎについて、$F''_i(x) \leftarrow \alpha_i F(x+r_i) + F'_i(x)$を計算する。この計算は、ｘについての多項式の組$F(x+r_i)$を多項式の組$F'_i(x)$によりマスクする操作に相当する。そして、検証者アルゴリズムＶは、多項式の組$F''_1, \cdots, F''_N$を検証者に送る。
【０２８３】
Ｓｔｅｐ．４：
検証者アルゴリズムＶは、ｉ＝１～Ｎのそれぞれについて、２つの検証パターンのうち、どちらの検証パターンを利用するかを選択する。次いで、検証者アルゴリズムＶは、選択した検証パターンを表す要求$d_i \in \{0, 1\}$（ｉ＝１～Ｎ）を証明者に送る。
【０２８４】
Ｓｔｅｐ．５：
証明者アルゴリズムＰは、検証者から受けた要求d_iに応じて検証者に送り返す情報σ_iを生成する。ここで、証明者アルゴリズムＰは、ｉ＝１～Ｎについて、以下の（処理１）又は（処理２）を実行する。（処理１）$d_i = 0$の場合、証明者アルゴリズムＰは、情報$\sigma_i = w_i$を生成する。（処理２）$d_i = 1$の場合、証明者アルゴリズムＰは、情報$\sigma_i = z_i$を生成する。上記（処理１）又は（処理２）の判定及び処理が実行された後、情報σ_i（ｉ＝１～Ｎ）は、証明者アルゴリズムＰにより検証者に送られる。
【０２８５】
Ｓｔｅｐ．６：
検証者アルゴリズムＶは、証明者から受け取った情報σ_i（ｉ＝１～Ｎ）を利用して以下の検証処理を実行する。なお、以下の処理は、ｉ＝１～Ｎについて実行される。
【０２８６】
$d_i = 0$の場合、検証者アルゴリズムＶは、$(r'_i, F'''_i) \leftarrow G(\sigma_i)$を計算する。そして、検証者アルゴリズムＶは、$c_{2,i} = H_2(\sigma_i)$の等号が成り立つか否かを検証する。また、検証者アルゴリズムＶは、$F''_i(x) = \alpha_i F(x+r'_i) + F'''_i(x)$の等号が成り立つか否かを検証する。検証者アルゴリズムＶは、これらの検証が全て成功した場合に認証成功を示す値１を出力し、検証に失敗があった場合に認証失敗を示す値０を出力する。
【０２８７】
$d_i = 1$の場合、検証者アルゴリズムＶは、$z'_i \leftarrow \sigma_i$を計算する。そして、検証者アルゴリズムＶは、$c_{1,i} = H_1(F'''_i(z'_i) - \alpha_i y, z_i)$の等号が成り立つか否かを検証する。検証者アルゴリズムＶは、この検証が成功した場合に認証成功を示す値１を出力し、検証に失敗があった場合に認証失敗を示す値０を出力する。
【０２８８】
以上、本手法の対話プロトコルを並列的に実行する方法について説明した。上記のように、本手法の対話プロトコルを繰り返し実行することにより、偽証が成功する確率を無視できる程度まで低減させることが可能になる。なお、拡張方式についても同様にして並列化することが可能である。拡張方式について並列化した対話プロトコルのアルゴリズム構成を図１５に示した。
【０２８９】
なお、上記のＳｔｅｐ．１の後、$(c_{1,1}, c_{1,2}, \cdots, c_{N,1}, c_{N,2})$を検証者に送信する代わりに，そのハッシュ値$H(c_{1,1}, c_{1,2}, \cdots, c_{N,1}, c_{N,2})$にまとめて送信してもよい．この構成を適用すると、１パス目で送るメッセージが１つのハッシュ値だけとなり、大幅に通信量を削減することが可能になる。但し、このハッシュ値、及び証明者から送られるチャレンジに対する回答から復元できないハッシュ値については、回答と併せて証明者から送ってもらうようにする。この構成によると、Ｎ回の並列的繰り返し構成の場合、送るべき情報の数をＮ－１個削減することが可能になる。

【０２９０】
（拡張方式に係る並列化アルゴリズムについて）
以下、図１５を参照しながら、拡張方式に係る並列化アルゴリズムの内容について説明する。なお、鍵生成アルゴリズムＧｅｎの構成については本手法に係る並列化アルゴリズムと同じであるため、詳細な説明を省略する。
【０２９１】
Ｓｔｅｐ．１：
まず、証明者アルゴリズムＰは、ｉ＝１～Ｎについて、以下の処理（１）～処理（５）を実行する。（処理（１））証明者アルゴリズムＰは、任意に数w_iを選択する。（処理（２））証明者アルゴリズムＰは、擬似乱数生成器Ｇに数w_iを適用してベクトル$r_i \in K^n$と多項式の組$F'_i(x)$を生成する。つまり、証明者アルゴリズムＰは、$(r_i, F'_i) \leftarrow G(w_i)$を計算する。（処理（３））証明者アルゴリズムＰは、$z_i \leftarrow s - r_i$を計算する。この計算は、秘密鍵ｓをベクトルr_iによりマスクする操作に相当する。
【０２９２】
Ｓｔｅｐ．１（続き）：
（処理（４））証明者アルゴリズムＰは、$F'_i(z_i)$とz_iのハッシュ値$c_{1,i}$を生成する。つまり、証明者アルゴリズムＰは、$c_{1,i} \leftarrow H_1(F'_i(z_i), z_i)$を計算する。（処理（５））証明者アルゴリズムＰは、数w_iのハッシュ値$c_{2,i}$を生成する。つまり、証明者アルゴリズムＰは、$c_{2,i} \leftarrow H_2(w_i)$を計算する。
【０２９３】
ｉ＝１～Ｎについて、上記の処理（１）～（５）が実行された後、Ｓｔｅｐ．１で生成されたメッセージ$(c_{1,i}, c_{2,i})$（ｉ＝１～Ｎ）は、検証者に送られる。
【０２９４】
Ｓｔｅｐ．２：
検証者アルゴリズムＶは、ｑ通り存在する環Ｋの元からＮ個の乱数$\alpha_1, \cdots, \alpha_N$を選択する。そして、検証者アルゴリズムＶは、選択した乱数$\alpha_1, \cdots, \alpha_N$を証明者に送る。
【０２９５】
Ｓｔｅｐ．３：
検証者アルゴリズムＶは、ｉ－１～Ｎについて、$F''_i(x) \leftarrow \alpha_i F(x + r_i) + F'_i(x)$を計算する。この計算は、ｘについての多項式の組$F(x + r_i)$を多項式の組$F'_i(x)$によりマスクする操作に相当する。次いで、検証者アルゴリズムＶは、多項式の組$F''_1, \cdots, F''_N$のハッシュ値c_3を生成する。つまり、証明者アルゴリズムＰは、$c_3 \leftarrow H_3(F''_1, \cdots, F''_N)$を計算する。なお、上記の$H_3(\cdots)$は、ハッシュ関数である。また、ハッシュ値$c_3$はメッセージである。
【０２９６】
Ｓｔｅｐ．３で生成されたメッセージc_3は、検証者に送られる。
【０２９７】
Ｓｔｅｐ．４：
検証者アルゴリズムＶは、ｉ＝１～Ｎのそれぞれについて、２つの検証パターンのうち、どちらの検証パターンを利用するかを選択する。次いで、検証者アルゴリズムＶは、選択した検証パターンを表す要求$d_i \in \{0, 1\}$（ｉ＝１～Ｎ）を証明者に送る。
【０２９８】
Ｓｔｅｐ．５：
証明者アルゴリズムＰは、検証者から受けた要求d_iに応じて検証者に送り返す情報σ_iを生成する。ここで、証明者アルゴリズムＰは、ｉ＝１～Ｎについて、以下の（処理１）又は（処理２）を実行する。（処理１）$d_i = 0$の場合、証明者アルゴリズムＰは、情報$\sigma_i = w_i$を生成する。（処理２）$d_i = 1$の場合、証明者アルゴリズムＰは、情報$\sigma_i = (z_i, F''_i)$を生成する。上記（処理１）又は（処理２）の判定及び処理が実行

された後、情報σ_i（ｉ＝１～Ｎ）は、証明者アルゴリズムＰにより検証者に送られる。

【０２９９】

Ｓｔｅｐ．６：

検証者アルゴリズムＶは、証明者から受け取った情報σ_i（ｉ＝１～Ｎ）を利用して以下の検証処理を実行する。なお、以下の処理は、ｉ＝１～Ｎについて実行される。

【０３００】

$d_i=0$の場合、検証者アルゴリズムＶは、$(r'_i, F'''_i) \leftarrow G(\sigma_i)$を計算する。さらに、検証者アルゴリズムＶは、$F''''_i \leftarrow \alpha_i F(x+r'_i)+F'''_i(x)$を計算する。そして、検証者アルゴリズムＶは、$c_{2,i}=H_2(\sigma_i)$の等号が成り立つか否かを検証する。また、検証者アルゴリズムＶは、$c_3=H_3(F''''_1, \cdots, F''''_N)$の等号が成り立つか否かを検証する。検証者アルゴリズムＶは、これらの検証が全て成功した場合に認証成功を示す値１を出力し、検証に失敗があった場合に認証失敗を示す値０を出力する。

【０３０１】

$d_i=1$の場合、検証者アルゴリズムＶは、$(z'_i, F''''_i) \leftarrow \sigma_i$とする。そして、検証者アルゴリズムＶは、$c_{1,i}=H_1(F''''_i(z'_i)-\alpha_i y, z'_i)$の等号が成り立つか否かを検証する。さらに、検証者アルゴリズムＶは、$c_3=H_3(F''''_1, \cdots, F''''_N)$の等号が成り立つか否かを検証する。検証者アルゴリズムＶは、これらの検証が全て成功した場合に認証成功を示す値１を出力し、検証に失敗があった場合に認証失敗を示す値０を出力する。

【０３０２】

以上、拡張方式に係る並列化アルゴリズムの内容について説明した。

【０３０３】

（好適なパラメータの設定方法について）

さて、上記の第１実施形態に係る対話プロトコルと同様、本実施形態に係る対話プロトコルは、受動的攻撃に対する安全性レベルを保証している。しかし、この対話プロトコルを並列的に繰り返し実行する上記の方法を適用した場合、能動的攻撃に対する安全性レベルが確実に保証されているということを保証するには、以下で説明するような条件が必要になる。

【０３０４】

上記の対話プロトコルは、１組の鍵ペア（公開鍵ｙ、秘密鍵ｓ）を用いて「ｙについてｙ＝Ｆ（ｓ）となるｓを知っていること」を証明者が検証者に対して証明する方式であった。そのため、検証で受理される対話を行った場合、「対話の際に証明者がｓを用いた」という情報を検証者に知られてしまう可能性が否定できない。また、上記のＦには衝突困難性が保証されてない。そのため、上記の対話プロトコルを並列的に繰り返し実行する場合について、能動的攻撃に対する安全性レベルが確実に保証されているということを無条件で証明することは難しいのである。

【０３０５】

そこで、本件発明者は、検証で受理される対話を行った場合においても、「対話の際に証明者がｓを用いた」という情報を検証者に知られないようにする方法について検討した。そして、本件発明者は、上記の対話プロトコルを並列的に繰り返し実行する場合においても、能動的攻撃に対する安全性を保証することが可能になる方法を考案した。この方法は、公開鍵として用いるｎ変数多次多項式$f_1, \cdots, f_m$のｍをその変数の数ｎよりも十分に小さな値に設定するというものである。例えば、$2^{m-n} \ll 1$となるようにｍ及びｎが設定される（例えば、ｎ＝１６０、ｍ＝８０の場合$2^{-80} \ll 1$である。）。

【０３０６】

上記のような多次多変数方程式の求解問題に対する解答の困難性に安全性の根拠をおく方式において、秘密鍵s_1と、それに対応する公開鍵ｐｋとが与えられても、その公開鍵ｐｋに対応する別の秘密鍵s_2を生成することは難しい。そのため、公開鍵ｐｋに対する秘密鍵ｓが２つ以上存在することを保証すれば、検証で受理される対話を行った場合にお

いても、「対話の際に証明者がｓを用いた」という情報を検証者に知られないようにすることが可能になる。つまり、この保証が成り立てば、対話プロトコルを並列的に繰り返し実行する場合においても、能動的攻撃に対する安全性を保証することが可能になる。

【０３０７】

図２８を参照しながら、ｍ本のｎ次多変数多項式で構成される関数Ｆ：$K^n \rightarrow K^m$について考えると（但し、ｎ＞ｍ）、２つ目の原像を持たない定義域の要素数は、最大で$|K|^m - 1$個しか存在しない。そのため、$|K|^{m-n}$を十分に小さくすると、２つ目の原像を持たない定義域の要素が選択される確率を無視できる程度まで小さくすることができる。つまり、ｎ変数多次多項式$f_1, \cdots, f_m$の数ｍがその変数の数ｎよりも十分に小さな値に設定されていれば、公開鍵ｐｋに対して２つ以上の秘密鍵ｓが存在することを保証できる。その結果、検証で受理される対話を行った場合においても、「対話の際に証明者がｓを用いた」という情報を検証者に知られないようにすることが可能になり、対話プロトコルを並列的に繰り返し実行する場合においても、能動的攻撃に対する安全性が保証される。

【０３０８】

以上説明したように、ｎ変数多次多項式$f_1, \cdots, f_m$の数ｍがその変数の数ｎよりも小さな値（ｎ＞ｍ；好ましくは$2^{m-n} \ll 1$）に設定する方法を適用することで、対話プロトコルを並列的に繰り返し実行する場合の安全性を向上させることが可能になる。

【０３０９】

［３－４：非対話型アルゴリズム］

これまで５パスの公開鍵認証方式について説明してきた。しかし、本手法においては、検証者から証明者に送られる情報が実際には単なる乱数だけであるため、１パスの公開鍵認証方式（以下、非対話型方式）に変形することができる。なお、非対話型方式に係る各アルゴリズムの内容を図１６に示した。以下、図１６を参照しながら、非対話型方式に係る各アルゴリズムの内容について説明する。

【０３１０】

（鍵生成アルゴリズムＧｅｎ）

鍵生成アルゴリズムＧｅｎは、環Ｋ上で定義されるｍ本のｎ変数多次多項式$f_1(x_1, \cdots, x_n), \cdots, f_m(x_1, \cdots, x_n)$と、ベクトル$s = (s_1, \cdots, s_n) \in K^n$を生成する。次に、鍵生成アルゴリズムＧｅｎは、$y = (y_1, \cdots, y_m) \leftarrow (f_1(s), \cdots, f_m(s))$を計算する。そして、鍵生成アルゴリズムＧｅｎは、$(f_1, \cdots, f_m, y)$を公開鍵ｐｋに設定し、ｓを秘密鍵に設定する。なお、以下では、ｎ変数のベクトル$(x_1, \cdots, x_n)$をｘと表記し、ｍ本のｎ変数多次多項式$(f_1(x), \cdots, f_m(x))$をＦ（ｘ）と表記する。

【０３１１】

（証明者アルゴリズムＰ、検証者アルゴリズムＶ）

次に、図１６を参照しながら、対話プロトコルにおける証明者アルゴリズムＰと検証者アルゴリズムＶによる処理について説明する。この対話プロトコルは、「証明者がｙ＝Ｆ（ｓ）を満たすｓを知っていること」をｓの情報を検証者に一切漏らさずに、検証者に証明させるものである。なお、鍵生成アルゴリズムＧｅｎにより生成された公開鍵ｐｋは、証明者と検証者の間で共有されているものとする。また、鍵生成アルゴリズムＧｅｎにより生成された秘密鍵ｓは、証明者により秘密に管理されているものとする。

【０３１２】

Ｓｔｅｐ．１：

まず、証明者アルゴリズムＰは、ｉ＝１～Ｎについて、以下の処理（１）～処理（５）を実行する。（処理（１））証明者アルゴリズムＰは、任意に数w_iを選択する。（処理（２））証明者アルゴリズムＰは、擬似乱数生成器Ｇに数w_iを適用してベクトル$r_i \in K^n$と多項式の組$F'_i(x)$を生成する。つまり、証明者アルゴリズムＰは、$(r_i, F'_i) \leftarrow G(w_i)$を計算する。（処理（３））証明者アルゴリズムＰは、$z_i \leftarrow s - r_i$を計算する。この計算は、秘密鍵ｓをベクトルr_iによりマスクする操作に相当する

。

【０３１３】

Ｓｔｅｐ．１（続き）：

（処理（４））証明者アルゴリズムＰは、$F_i(z_i)$とz_iのハッシュ値$c_{1,i}$を生成する。つまり、証明者アルゴリズムＰは、$c_{1,i} \leftarrow H_1(F_i(z_i), z_i)$を計算する。（処理（５））証明者アルゴリズムＰは、数w_iのハッシュ値$c_{2,i}$を生成する。つまり、証明者アルゴリズムＰは、$c_{2,i} \leftarrow H_2(w_i)$を計算する。

【０３１４】

Ｓｔｅｐ．１（続き）：

次に、検証者アルゴリズムＶは、乱数R_A，R_Bを選択する。そして、検証者アルゴリズムＶは、乱数R_Aと上記の処理（４）及び（５）で計算されたハッシュ値$c_{1,i}$，$c_{2,i}$をハッシュ関数H_Aに適用してハッシュ値α_1，…，α_Nを生成する。つまり、検証者アルゴリズムＶは、$(\alpha_1, \cdots, \alpha_N) \leftarrow H_A(R_A, c_{1,1}, \cdots, c_{2,N})$を計算する。

【０３１５】

Ｓｔｅｐ．１（続き）：

次に、検証者アルゴリズムＶは、ｉ＝１～Ｎについて、以下の処理（１）及び（２）を実行する。（処理（１））証明者アルゴリズムＰは、$F''_i(x) \leftarrow \alpha_i F(x + r_i) + F'_i(x)$を計算する。この計算は、ｘについての多項式の組$F(x + r_i)$を多項式の組$F'_i(x)$によりマスクする操作に相当する。（処理（２））証明者アルゴリズムＰは、ｉ＝１～Ｎについて、乱数R_A，R_B、ハッシュ値（$c_{1,i}$、$c_{2,i}$）、α_i，F''_iをハッシュ関数H_Bに適用して$d = (d_1, \cdots, d_N)$を生成する。つまり、証明者アルゴリズムＰは、$(d_1, \cdots, d_N) \leftarrow H_B(R_A, R_B, c_{1,1}, \cdots, c_{2,N}, \alpha_1, \cdots, \alpha_N, F''_1, \cdots, F''_N)$を計算する。

【０３１６】

Ｓｔｅｐ．１（続き）：

次に、証明者アルゴリズムＰは、生成したd_iに応じて検証者に送り返す情報σ_iを生成する。ここで、証明者アルゴリズムＰは、ｉ＝１～Ｎについて、以下の（処理１）又は（処理２）を実行する。（処理１）$d_i = 0$の場合、証明者アルゴリズムＰは、情報$\sigma_i = w_i$を生成する。（処理２）$d_i = 1$の場合、証明者アルゴリズムＰは、情報$\sigma_i = z_i$を生成する。上記（処理１）又は（処理２）の判定及び処理が実行された後、R_A、R_B、$c_{1,i}$、$c_{2,i}$、σ_i（ｉ＝１～Ｎ）は、証明者アルゴリズムＰにより検証者に送られる。

【０３１７】

Ｓｔｅｐ．２：

検証者アルゴリズムＶは、まず、証明者から受け取ったR_A、$c_{1,i}$、$c_{2,i}$をハッシュ関数H_Aに適用してα_iを生成する。つまり、検証者アルゴリズムＶは、$(\alpha_1, \cdots, \alpha_N) \leftarrow H_A(R_A, c_{1,1}, \cdots, c_{2,N})$を計算する。次いで、検証者アルゴリズムＶは、$d = (d_1, \cdots, d_N) \leftarrow H_B(R_A, R_B, c_{1,1}, \cdots, c_{2,N}, \alpha_1, \cdots, \alpha_N, F''_1, \cdots, F''_N)$を計算する。次いで、検証者アルゴリズムＶは、情報$\sigma_i$（ｉ＝１～Ｎ）を利用して以下の検証処理を実行する。なお、以下の処理は、ｉ＝１～Ｎについて実行される。

【０３１８】

$d_i = 0$の場合、検証者アルゴリズムＶは、$(r'_i, F'''_i) \leftarrow G(\sigma_i)$を計算する。そして、検証者アルゴリズムＶは、$c_{2,i} = H_2(\sigma_i)$の等号が成り立つか否かを検証する。また、検証者アルゴリズムＶは、$F''_i(x) = \alpha_i F(x + r'_i) + F'''_i(x)$の等号が成り立つか否かを検証する。検証者アルゴリズムＶは、これらの検証が全て成功した場合に認証成功を示す値１を出力し、検証に失敗があった場合に認証失敗を示す値０を出力する。

【０３１９】

$d_i=1$の場合、検証者アルゴリズムVは、$z'_i \leftarrow \sigma_i$を計算する。そして、検証者アルゴリズムVは、$c_{1,i}=H_1(F'''_i(z'_i)-\alpha_i y, z'_i)$の等号が成り立つか否かを検証する。検証者アルゴリズムVは、この検証が成功した場合に認証成功を示す値1を出力し、検証に失敗があった場合に認証失敗を示す値0を出力する。

【0320】

以上、非対話型方式に係る各アルゴリズムの内容について説明した。なお、理想的なハッシュ関数H_A、H_Bを仮定すれば、ハッシュ値α_i, d_iがランダムに振る舞うため、証明者に都合の良いハッシュ値α_i, d_iが出ることはない。そのため、上記のような変形を行っても、十分な安全性が確保される。なお、このような変形は、拡張方式などに対しても同様に適用することが可能である。

【0321】

［3－5：電子署名方式への変形］

ここで、本手法を電子署名方式へと変形する方法について説明する。なお、ここでは簡単のために、上記の非対話型方式を電子署名方式に変形する方法について述べる。上記の非対話型方式における証明者と検証者を電子署名方式における署名者と検証者に対応させると、証明者のみが検証者を納得させられるという点において、電子署名方式のモデルと近似していることが分かるであろう。こうした考えをもとに、非対話型方式に基づく電子署名方式のアルゴリズム構成について詳細に説明する。

【0322】

（鍵生成アルゴリズムＧｅｎ）

鍵生成アルゴリズムＧｅｎは、環K上で定義されるm本のn変数多次多項式$f_1(x_1, \cdots, x_n), \cdots, f_m(x_1, \cdots, x_n)$と、ベクトル$s=(s_1, \cdots, s_n) \in K^n$を生成する。次に、鍵生成アルゴリズムＧｅｎは、$y=(y_1, \cdots, y_m) \leftarrow (f_1(s), \cdots, f_m(s))$を計算する。そして、鍵生成アルゴリズムＧｅｎは、$(f_1, \cdots, f_m, y)$を公開鍵ｐｋに設定し、sを秘密鍵に設定する。なお、以下では、n変数のベクトル$(x_1, \cdots, x_n)$をxと表記し、m本のn変数多次多項式$(f_1(x), \cdots, f_m(x))$をF（x）と表記する。

【0323】

（署名生成アルゴリズムＳｉｇ）

署名生成アルゴリズムＳｉｇは、ｉ＝１～Ｎについて、以下の（処理１）～（処理１１）を実行する。なお、署名生成アルゴリズムＳｉｇには、署名鍵ｓｋ＝ｓと文書Ｍが入力されているものとする。

【0324】

（処理１）署名生成アルゴリズムＳｉｇは、任意に数w_iを選択する。（処理２）署名生成アルゴリズムＳｉｇは、擬似乱数生成器Ｇに数w_iを適用してベクトル$r_i \in K^n$とｎ変数多項式の組$F'_i(x)=(f_{1,1}'(x), \cdots, f_{m,i}'(x))$を生成する。つまり、署名生成アルゴリズムＳｉｇは、$(r_i, F'_i) \leftarrow G(w_i)$を計算する。（処理２）署名生成アルゴリズムＳｉｇは、$z_i \leftarrow s-r_i$を計算する。この計算は、秘密鍵ｓをベクトルr_iによりマスクする操作に相当する。

【0325】

（処理３）署名生成アルゴリズムＳｉｇは、$F''_i(z_i)$とz_iのハッシュ値$c_{1,i}$を生成する。つまり、署名生成アルゴリズムＳｉｇは、$c_{1,i} \leftarrow H_1(F''_i(z_i), z_i)$を計算する。（処理４）署名生成アルゴリズムＳｉｇは、数w'_iのハッシュ値$c_{2,i}$を生成する。つまり、署名生成アルゴリズムＳｉｇは、$c_{2,i} \leftarrow H_2(w'_i)$を計算する。なお、上記の$H_1$（…）、$H_2$（…）は、ハッシュ関数である。

【0326】

（処理４）署名生成アルゴリズムＳｉｇは、乱数R_Aを任意に選択する。（処理５）署名生成アルゴリズムＳｉｇは、$\alpha=(\alpha_1, \cdots, \alpha_N) \leftarrow H_A(R_A, M, c_{1,1}, \cdots, c_{2,N})$を計算する。（処理６）署名生成アルゴリズムＳｉｇは、ｉ＝１～Ｎについて、$F''_i(x)=\alpha_i F(x+r'_i)+F'_i(x)$を計算する。（処理７）署名生

成アルゴリズムＳｉｇは、乱数R_Bを任意に選択する。（処理８）署名生成アルゴリズムＳｉｇは、$d=(d_1,\cdots,d_N)\leftarrow H_B(R_B,M,c_{1,1},\cdots,c_{2,N},\alpha,F''_1,\cdots,F''_N)$を計算する。

【０３２７】

次に、署名生成アルゴリズムＳｉｇは、ｉ＝１～Ｎについて、d_iに応じた次の（処理９）又は（処理１０）を実行する。（処理９）$d_i=0$の場合、署名生成アルゴリズムＳｉｇは、$\sigma_i\leftarrow w_i$を計算する。（処理１０）$d_i=1$の場合、署名生成アルゴリズムＳｉｇは、$\sigma_i\leftarrow z_i$を計算する。

【０３２８】

（処理１１）ｉ＝１～Ｎについて、上記（処理９）又は（処理１０）の判定及び処理が実行された後、署名生成アルゴリズムＳｉｇは、電子署名$\sigma=(R_A,R_B,c_{1,1},\cdots,c_{2,N},\alpha_1,\cdots,\alpha_N,F''_1,\cdots,F''_N,\sigma_1,\cdots,\sigma_N)$を出力する。

【０３２９】

（署名検証アルゴリズムＶｅｒ）

署名検証アルゴリズムＶｅｒは、ｉ＝１～Ｎについて、以下の全ての検証を通過すれば、電子署名σを受理し、１つでも検証を通過しなかった場合には電子署名σを拒否する。なお、署名検証アルゴリズムＶｅｒには、電子署名σと文書Ｍが入力されているものとする。まず、署名検証アルゴリズムＶｅｒは、$\alpha=(\alpha_1,\cdots,\alpha_N)\leftarrow H_A(R_A,M,c_{1,1},\cdots,c_{3,N})$を計算する。次いで、署名検証アルゴリズムＶｅｒは、$d=(d_1,\cdots,d_N)\leftarrow H(R,M,c_{1,1},\cdots,c_{3,N},\alpha,F''_1,\cdots,F''_N)$を計算する。次いで、署名検証アルゴリズムＶｅｒは、ｉ＝１～Ｎについて、以下の（検証１）～（検証３）を実行する。

【０３３０】

（検証１）ｄｉ＝０の場合、署名検証アルゴリズムＶｅｒは、$(r'_i,F''_i)\leftarrow G(\sigma_i)$を計算する。そして、署名検証アルゴリズムＶｅｒは、$c_{2,i}=H_2(\sigma_i)$の等号が成り立つか否かを検証する。また、署名検証アルゴリズムＶｅｒは、$F''_i(x)=\alpha_i F(x+r'_i)+F'_i(x_1,\cdots,x_n)$の等号が成り立つか否かを検証する。署名検証アルゴリズムＶｅｒは、これらの検証が全て成功した場合に電子署名σの受理を示す値１を出力し、検証に失敗があった場合に電子署名σの拒否を示す値０を出力する。

【０３３１】

（検証２）$d_i=1$の場合、署名検証アルゴリズムＶｅｒは、$(z'_i,F'''_i)\leftarrow\sigma_i$を計算する。そして、署名検証アルゴリズムＶｅｒは、$c_{1,i}=H_1(F'''_i(z'_i),z'_i)$の等号が成り立つか否かを検証する。署名検証アルゴリズムＶｅｒは、この検証が成功した場合に電子署名σの受理を示す値１を出力し、検証に失敗があった場合に電子署名σの拒否を示す値０を出力する。

【０３３２】

以上、本手法に基づく電子署名方式の各アルゴリズム構成について説明した。なお、同様にして、上記の拡張方式に基づく電子署名方式を構築することも可能である。

【０３３３】

［３－６：具体例］

次に、図１７を参照しながら、本手法を実施する際に想定される具体的なアルゴリズム構成について説明する。図１７は、本手法の具体例を説明するための説明図である。

【０３３４】

（鍵生成アルゴリズムＧｅｎ）

鍵生成アルゴリズムＧｅｎは、環Ｋ上で定義されるｍ本のｎ変数２次多項式$f_1(x_1,\cdots,x_n),\cdots,f_m(x_1,\cdots,x_n)$と、ベクトル$s=(s_1,\cdots,s_n)\in K^n$を生成する。次に、鍵生成アルゴリズムＧｅｎは、$y=(y_1,\cdots,y_m)\leftarrow(f_1(s),\cdots,f_m(s))$を計算する。そして、鍵生成アルゴリズムＧｅｎは、$(f_1,\cdots,f_m,y)$を公開鍵ｐｋに設定し、ｓを秘密鍵に設定する。なお、以下では、ｎ変数のベ

クトル（x_1，…，x_n）をｘと表記し、ｍ本のｎ変数多次多項式（f_1（ｘ），…，f_m（ｘ））をＦ（ｘ）と表記する。

【０３３５】

（証明者アルゴリズムＰ、検証者アルゴリズムＶ）

次に、図１７を参照しながら、対話プロトコルにおける証明者アルゴリズムＰと検証者アルゴリズムＶによる処理について説明する。この対話プロトコルは、「証明者がｙ＝Ｆ（ｓ）を満たすｓを知っていること」をｓの情報を検証者に一切漏らさずに、検証者に証明させるものである。なお、鍵生成アルゴリズムＧｅｎにより生成された公開鍵ｐｋは、証明者と検証者の間で共有されているものとする。また、鍵生成アルゴリズムＧｅｎにより生成された秘密鍵ｓは、証明者により秘密に管理されているものとする。

【０３３６】

Ｓｔｅｐ．１：

まず、証明者アルゴリズムＰは、任意に数ｗを選択する。次いで、証明者アルゴリズムＰは、擬似乱数生成器Ｇに数ｗを適用してベクトルｒ∈K^nと、ｎ変数多項式の組Ｆ’（ｘ）＝（f'_1（ｘ），…，f'_m（ｘ））を生成する。つまり、証明者アルゴリズムＰは、（ｒ，Ｆ’）←Ｇ（ｗ）を計算する。次いで、証明者アルゴリズムＰは、ｚ←ｓ－ｒを計算する。この計算は、秘密鍵ｓをベクトルｒによりマスクする操作に相当する。但し、多項式f'_i（ｘ）は、下記の式（２０）のように表現される。

【０３３７】

【数１４】

$$f'_i(x) = \sum_j b'_{i,j} x_j + c'_i$$

…（２０）

【０３３８】

Ｓｔｅｐ．１（続き）：

次いで、証明者アルゴリズムＰは、Ｆ’（ｚ）とｚのハッシュ値c_1を生成する。つまり、証明者アルゴリズムＰは、c_1←H_1（Ｆ’（ｚ），ｚ）を計算する。また、証明者アルゴリズムＰは、数ｗのハッシュ値c_2を生成する。つまり、証明者アルゴリズムＰは、c_2←H_2（ｗ）を計算する。なお、上記のH_1（…）、H_2（…）は、ハッシュ関数である。また、ハッシュ値（c_1，c_2）はメッセージである。

【０３３９】

Ｓｔｅｐ．１で生成されたメッセージ（c_1，c_2）は、検証者に送られる。

【０３４０】

Ｓｔｅｐ．２：

検証者アルゴリズムＶは、ｑ通り存在する環Ｋの元から１つの乱数αを選択する。そして、検証者アルゴリズムＶは、選択した乱数αを証明者に送る。

【０３４１】

Ｓｔｅｐ．３：

証明者アルゴリズムＰは、Ｆ”（ｘ）←αＦ（ｘ＋ｒ）＋Ｆ’（ｘ）を計算する。この計算は、ｘについての多項式の組Ｆ（ｘ＋ｒ）を多項式の組Ｆ’（ｘ）によりマスクする操作に相当する。

【０３４２】

Ｓｔｅｐ．３で生成された多項式Ｆ”は、検証者に送られる。

【０３４３】

Ｓｔｅｐ．４：

検証者アルゴリズムＶは、２つの検証パターンのうち、どちらの検証パターンを利用するかを選択する。次いで、検証者アルゴリズムＶは、選択した検証パターンを表す要求ｄ

∈｛０，１｝を証明者に送る。

【０３４４】

Ｓｔｅｐ．５：

証明者アルゴリズムＰは、検証者から受けた要求ｄに応じて検証者に送り返す情報σを生成する。もしｄ＝０の場合、証明者アルゴリズムＰは、情報σ＝ｗを生成する。また、ｄ＝１の場合、証明者アルゴリズムＰは、情報σ＝ｚを生成する。このようにして生成された情報σは、証明者アルゴリズムＰにより検証者に送られる。

【０３４５】

Ｓｔｅｐ．６：

検証者アルゴリズムＶは、証明者から受け取った情報σを利用して以下の検証処理を実行する。

【０３４６】

ｄ＝０の場合、検証者アルゴリズムＶは、（ｒ’，Ｆ'''）←Ｇ（σ）を計算する。そして、検証者アルゴリズムＶは、$c_2 = H_2(\sigma)$の等号が成り立つか否かを検証する。また、検証者アルゴリズムＶは、Ｆ”（ｘ）＝αＦ（ｘ＋ｒ’）＋Ｆ'''（ｘ）の等号が成り立つか否かを検証する。検証者アルゴリズムＶは、これらの検証が全て成功した場合に認証成功を示す値１を出力し、検証に失敗があった場合に認証失敗を示す値０を出力する。

【０３４７】

ｄ＝１の場合、検証者アルゴリズムＶは、ｚ’←σとする。そして、検証者アルゴリズムＶは、$c_1 = H_1(F''(z') - \alpha y, z')$の等号が成り立つか否かを検証する。検証者アルゴリズムＶは、この検証が成功した場合に認証成功を示す値１を出力し、検証に失敗があった場合に認証失敗を示す値０を出力する。

【０３４８】

以上、本手法を実施する際に想定される具体的なアルゴリズム構成について説明した。

【０３４９】

［３－７：効率的なアルゴリズム］

ここで、上記の第１実施形態と同様、効率の良いアルゴリズムについて考える。例えば、ｘについての多項式の組Ｆ（ｘ＋ｒ）をマスクするために用いた多項式の組Ｆ’（ｘ）を、２つのベクトル$t \in K^n$、$e \in K^m$を用いてＦ’（ｘ）＝F_b（ｘ、ｔ）＋ｅのように表現する方法について考える。この方法を用いると、ｘについての多項式の組Ｆ（ｘ＋ｒ）は、下記の式（２１）のように表現される。

【０３５０】

【数１５】

$$\begin{aligned}&\alpha F(x+r)+F'(x)\\&=\alpha F(x)+\alpha F(r)+\alpha F_b(x,r)-\alpha c+F_b(x,t)+e\\&=\alpha F(x)+F_b(x,\alpha r+t)+\alpha(F(r)-c)+e\end{aligned}$$

…（２１）

【０３５１】

そのため、ｔ’＝αｒ＋ｔ，ｅ’＝α（Ｆ（ｒ）－ｃ）＋ｅとすれば、マスク後のｘに関する多項式の組Ｆ”（ｘ）も、２つのベクトル$t' \in K^n$、$e \in K^m$により表現することができる。これらの理由から、Ｆ’をＦ’（ｘ）＝F_b（ｘ，ｔ）＋ｅと設定すれば、Ｆ’とＦ”の表現がK^n上のベクトルとK^m上のベクトルによる表現されることになり、通信に必要なデータサイズを大幅に削減することが可能になる。

【０３５２】

また、上記の変形により、Ｆ”（又はＦ’）からｒに関する情報が一切漏れることはな

い。例えば、ｅ’，ｔ’（又はｅ，ｔ）を与えられても、ｅ，ｔ（又はｅ’，ｔ’）を知らない限り、ｒの情報を一切知ることはできない。従って、上記の変形を施した本手法は、零知識性が担保された公開鍵認証方式である。以下、図１８～図２５を参照しながら、上記の変形を施した本手法に係る効率的なアルゴリズムについて、より詳細に説明する。なお、鍵生成アルゴリズムＧｅｎの構成については先に説明した本手法と実質的に同じであるため、詳細な説明を省略する。

【０３５３】

Ｓｔｅｐ．１：

まず、証明者アルゴリズムＰは、任意に数ｗを選択する。次いで、証明者アルゴリズムＰは、擬似乱数生成器Ｇに数ｗを適用してベクトル$r \in K^n$、$t \in K^n$、$e \in K^m$を生成する。つまり、証明者アルゴリズムＰは、（ｒ，ｔ，ｅ）←Ｇ（ｗ）を計算する。次いで、証明者アルゴリズムＰは、ｚ←ｓ－ｒを計算する。この計算は、秘密鍵ｓをベクトルｒによりマスクする操作に相当する。

【０３５４】

Ｓｔｅｐ．１（続き）：

次いで、証明者アルゴリズムＰは、F_b（ｚ，ｔ）＋ｅとｚのハッシュ値c_1を生成する。つまり、証明者アルゴリズムＰは、$c_1 \leftarrow H_1$（F_b（ｚ，ｔ）＋ｅ，ｚ）を計算する。また、証明者アルゴリズムＰは、数ｗのハッシュ値c_2を生成する。つまり、証明者アルゴリズムＰは、$c_2 \leftarrow H_2$（ｗ）を計算する。なお、上記のH_1（…）、H_2（…）は、ハッシュ関数である。また、ハッシュ値（c_1，c_2）はメッセージである。

【０３５５】

Ｓｔｅｐ．１で生成されたメッセージ（c_1，c_2）は、検証者に送られる。

【０３５６】

Ｓｔｅｐ．２：

検証者アルゴリズムＶは、ｑ通り存在する環Ｋの元から１つの乱数αを選択する。そして、検証者アルゴリズムＶは、選択した乱数αを証明者に送る。

【０３５７】

Ｓｔｅｐ．３：

証明者アルゴリズムＰは、ｔ’←αｒ＋ｔを計算する。さらに、証明者アルゴリズムＰは、ｅ’←α（Ｆ（ｒ）－ｃ）＋ｅを計算する。そして、証明者アルゴリズムＰは、ｔ’，ｅ’を検証者に送る。

【０３５８】

Ｓｔｅｐ．４：

検証者アルゴリズムＶは、２つの検証パターンのうち、どちらの検証パターンを利用するかを選択する。次いで、検証者アルゴリズムＶは、選択した検証パターンを表す要求ｄ∈｛０，１｝を証明者に送る。

【０３５９】

Ｓｔｅｐ．５：

証明者アルゴリズムＰは、検証者から受けた要求ｄに応じて検証者に送り返す情報σを生成する。もしｄ＝０の場合、証明者アルゴリズムＰは、情報σ＝ｗを生成する。また、ｄ＝１の場合、証明者アルゴリズムＰは、情報σ＝ｚを生成する。このようにして生成された情報σは、証明者アルゴリズムＰにより検証者に送られる。

【０３６０】

Ｓｔｅｐ．６：

検証者アルゴリズムＶは、証明者から受け取った情報σを利用して以下の検証処理を実行する。

【０３６１】

ｄ＝０の場合、検証者アルゴリズムＶは、（ｒ’，ｔ”，ｅ”）←Ｇ（σ）を計算する。そして、検証者アルゴリズムＶは、$c_2 = H_2$（σ）の等号が成り立つか否かを検証する。また、検証者アルゴリズムＶは、ｔ’＝αｒ’＋ｔ”の等号が成り立つか否かを検証

する。さらに、検証者アルゴリズムＶは、ｅ’＝α（Ｆ（ｒ’）－ｃ）＋ｅ”の等号が成り立つか否かを検証する。検証者アルゴリズムＶは、これらの検証が全て成功した場合に認証成功を示す値１を出力し、検証に失敗があった場合に認証失敗を示す値０を出力する。

【０３６２】

ｄ＝１の場合、検証者アルゴリズムＶは、ｚ’←σとする。そして、検証者アルゴリズムＶは、$c_1 = H_1(\alpha F(z') - y + F_b(z', t') + e', z')$の等号が成り立つか否かを検証する。検証者アルゴリズムＶは、この検証が成功した場合に認証成功を示す値１を出力し、検証に失敗があった場合に認証失敗を示す値０を出力する。

【０３６３】

以上、本手法に係る効率的なアルゴリズム構成について説明した。この効率的なアルゴリズムを利用することにより、通信に必要なデータサイズを大幅に削減することができる。また、対話プロトコルの中で、ｘ＋ｒを多項式Ｆに代入し、ｘについて整理しなおす計算（Ｆ（ｘ＋ｒ）の計算）が必要なくなるため、計算効率も向上する。

【０３６４】

（効率的なアルゴリズムの変形例について）

さて、図１８に示した本手法の効率的なアルゴリズムは、その計算効率を維持しながら、図１９～図２５に示すようなアルゴリズムに変形することが可能である。

【０３６５】

（変形例Ａ：図１９に示すアルゴリズム）

例えば、図１８に示した効率的なアルゴリズムのＳｔｅｐ．３において、ｅ’←α（Ｆ（ｒ）－ｃ）＋ｅという計算を行っているが、この計算を図１９に示すようにｅ’←αＦ（ｒ）＋ｅという計算に変形してもよい。但し、この変形を行う場合には、Ｓｔｅｐ．６において検証者が行う検証内容の一部を変形する必要がある。具体的には、Ｓｔｅｐ．６においてｄ＝０の場合に検証者が行う検証内容のうち、ｅ’＝α（Ｆ（ｒ’）－ｃ）＋ｅ”の検証をｅ’＝αＦ（ｒ’）＋ｅ”の検証に置き換える必要がある。また、Ｓｔｅｐ．６においてｄ＝１の場合に検証者が行う検証内容のうち、$c_1 = H_1(\alpha(F(z') - y) + F_b(z', t''') + e''', z')$の検証を$c_1 = H_1(\alpha(F(z') - c - y) + F_b(z', t''') + e''', z')$の検証に置き換える必要がある。

【０３６６】

（変形例Ｂ：図２０に示すアルゴリズム）

また、図１９に示したアルゴリズムのＳｔｅｐ．５において、ｄ＝０の場合にσをｗに設定しているが、ｄ＝０の場合に設定されるσは、（ｔ’，ｅ’’）に併せて用いることで（ｒ，ｔ，ｅ）が復元できるような情報であればどのような情報でもよい。例えば、図２０に示すように、Ｓｔｅｐ．５においてｄ＝０の場合に設定されるσの内容をｒにしてもよい。但し、この変形を行う場合には、図１９に示したアルゴリズムのＳｔｅｐ．１における計算$c_2 \leftarrow H_2(w)$を$c_2 \leftarrow H_2(r, t, e)$に変形する必要がある。また、Ｓｔｅｐ．６においてｄ＝０の場合に検証者が行う検証内容を$c_2 = H_2(r, t' - \alpha r, e' - \alpha F(r))$の検証に置き換える必要がある。

【０３６７】

（変形例Ｃ：図２１に示すアルゴリズム）

また、図２０に示したアルゴリズムのＳｔｅｐ．３において、ｔ’←αｒ＋ｔという計算を行っているが、この計算を図２１に示すようにｔ’←α（ｒ＋ｔ）という計算に変形してもよい。但し、この変形を行う場合には、図２０に示したアルゴリズムのＳｔｅｐ．６においてｄ＝０の場合に検証者が行う検証内容を$c_2 = H_2(r, \alpha^{-1} t' - r, e' - \alpha F(r))$の検証に置き換える必要がある。

【０３６８】

（変形例Ｄ：図２２に示すアルゴリズム）

また、図２０に示したアルゴリズムのＳｔｅｐ．３において、ｅ’←αＦ（ｒ）＋ｅと

いう計算を行っているが、この計算を図２２に示すようにｅ’←α（Ｆ（ｒ）＋ｅ）という計算に変形してもよい。但し、この変形を行う場合には、図２０に示したアルゴリズムのＳｔｅｐ．６においてｄ＝０の場合に検証者が行う検証内容を$c_2=H_2$（ｒ，ｔ’－αｒ，ｅ’－α^{-1}ｅ’－Ｆ（ｒ））の検証に置き換える必要がある。

【０３６９】

（変形例Ｅ：図２３に示すアルゴリズム）

また、図２０に示したアルゴリズムのＳｔｅｐ．５において、ｄ＝０の場合にσをｒに設定しているが、ｄ＝０の場合に設定されるσは、（ｔ’，ｅ’’）に併せて用いることで（ｒ，ｔ，ｅ）が復元できるような情報であればどのような情報でもよい。例えば、図２３に示すように、Ｓｔｅｐ．５においてｄ＝０の場合に設定されるσの内容をｔにしてもよい。但し、この変形を行う場合には、Ｓｔｅｐ．２においてαがα∈$_R$Ｋ＼｛０｝から選ばれるようにすると共に、Ｓｔｅｐ．６においてｄ＝０の場合に検証者が行う検証内容を$c_2=H_2$（α^{-1}（ｔ’－ｔ），ｔ，ｅ’－αＦ（α^{-1}（ｔ’－ｔ））の検証に置き換える必要がある。

【０３７０】

（変形例Ｆ：図２４に示すアルゴリズム）

また、図２３に示したアルゴリズムのＳｔｅｐ．３において、ｔ’←αｒ＋ｔという計算を行っているが、この計算を図２４に示すようにｔ’←α（ｒ＋ｔ）という計算に変形してもよい。但し、この変形を行う場合には、図２３に示したアルゴリズムのＳｔｅｐ．６においてｄ＝０の場合に検証者が行う検証内容を$c_2=H_2$（α^{-1}ｔ’－ｔ，ｔ，ｅ’－αＦ（α^{-1}ｔ’－ｔ））の検証に置き換える必要がある。

【０３７１】

（変形例Ｇ：図２５に示すアルゴリズム）

また、図２３に示したアルゴリズムのＳｔｅｐ．３において、ｅ’←αＦ（ｒ）＋ｅという計算を行っているが、この計算を図２５に示すようにｅ’←α（Ｆ（ｒ）＋ｅ）という計算に変形してもよい。但し、この変形を行う場合には、図２３に示したアルゴリズムのＳｔｅｐ．６においてｄ＝０の場合に検証者が行う検証内容を$c_2=H_2$（α^{-1}（ｔ’－ｔ），ｔ，α^{-1}ｅ’－αＦ（α^{-1}（ｔ’－ｔ））の検証に置き換える必要がある。

【０３７２】

上記のように、Ｓｔｅｐ．３において証明者が行うｅ’←α（Ｆ（ｒ）－ｃ）＋ｅという計算をｅ’←αＦ（ｒ）＋ｅという計算に変更することができる（変形例Ａ）。また、Ｓｔｅｐ．５においてｄ＝０の場合に証明者が設定するσの内容をｒやｔなどに変更することができる（変形例Ｂ、Ｅ）。さらに、Ｓｔｅｐ．３において証明者が行うｔ’←αｒ＋ｔという計算をｔ’←α（ｒ＋ｔ）という計算に変更することができる（変形例Ｃ、Ｆ）。また、Ｓｔｅｐ．３において証明者が行うｅ’←αＦ（ｒ）＋ｅという計算をｅ’←α（Ｆ（ｒ）＋ｅ）という計算に変更することができる（変形例Ｄ、Ｇ）。変形例Ａに係る変形、変形例Ｂ、Ｅに係る変形、変形例Ｃ、Ｆに係る変形、及び変形例Ｄ、Ｇに係る変形は、それぞれ単独で図１８に示した効率的なアルゴリズムに適用することもできるし、複数の変形を組み合わせて図１８に示した効率的なアルゴリズムに適用することもできる。

【０３７３】

以上、本発明の第２実施形態について説明した。なお、多変数連立方程式の形式については、上記の第１実施形態と同様である。

【０３７４】

＜４：効率的なアルゴリズムの一般化＞

ところで、上記の第１及び第２実施形態に係る効率的なアルゴリズムは、下記の式（２２）で表現される２次の多変数多項式を公開鍵（又はシステムパラメータ）として利用する構成であった。しかし、上記の効率的なアルゴリズムは、位数$q=p^k$の体上で定義される３次以上の多次多変数多項式（下記の式（２３）を参照）を公開鍵（又はシステムパ

ラメータ）として利用する構成に拡張することができる。

【０３７５】

【数１６】

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

…（２２）

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

…（２３）

【０３７６】

上記の第１及び第２実施形態に係る効率的なアルゴリズムにおける公開鍵として多変数多項式ｆ$_l$が利用可能となる条件は、下記の式（２４）が（ｘ$_1$，…，ｘ$_n$）及び（ｙ$_1$，…，ｙ$_n$）に対して双線形になることである。上記の式（２２）で表現される多変数多項式の場合、下記の式（２５）に示すように容易に双線形性を確認することができる（下線部がｘ$_i$、ｙ$_i$のそれぞれについて線形となっている）。また、上記の式（２３）で表現される多変数多項式の場合についても同様にして、下記の式（２６）に示すように双線形性を確認することができる。但し、下記の式（２６）の下線部は、位数ｐの体ＧＦ（ｐ）上における双線形性を示している。そのため、上記の第２実施形態に係る効率的なアルゴリズムの公開鍵として上記の式（２３）で表現される多変数多項式を利用する場合、当該アルゴリズムのＳｔｅｐ．２の後に検証者が送信するチャレンジαは、ＧＦ（ｐ）の要素に限定する必要がある。

【０３７７】

【数１７】

$$f_l(x_1+y_1,\cdots,x_n+y_n)-f_l(x_1,\cdots,x_n)-f_l(y_1,\cdots,y_n)+f_l(0,\cdots,0)$$

…（２４）

$$\begin{aligned}
&f_l(x_1+y_1,\cdots,x_n+y_n)\\
&=\sum_{i=1}^{n}\sum_{j=1}^{n}a_{l,i,j}(x_i+y_i)(x_j+y_j)+\sum_{i=1}^{n}b_{l,i}(x_i+y_i)+c_l\\
&=\sum_{i=1}^{n}\sum_{j=1}^{n}a_{l,i,j}(x_ix_j+x_iy_j+y_ix_j+y_iy_j)+\sum_{i=1}^{n}b_{l,i}(x_i+y_i)+c_l\\
&=f_l(x_1,\cdots,x_n)+f_l(y_1,\cdots,y_n)+\sum_{i=1}^{n}\sum_{j=1}^{n}a_{l,i,j}\underline{(x_iy_j+y_ix_j)}-f_l(0,\cdots,0)
\end{aligned}$$

…（２５）

$$\begin{aligned}
&f_l(x_1+y_1,\cdots,x_n+y_n)\\
&=\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{s=0}^{k-1}\sum_{t=0}^{k-1}a_{l,i,j,s,t}(x_i+y_i)^{p^s}(x_j+y_j)^{p^t}+\sum_{i=1}^{n}\sum_{s=0}^{k-1}b_{l,i,s}(x_i+y_i)^{p^s}+c_l\\
&=\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{s=0}^{k-1}\sum_{t=0}^{k-1}a_{l,i,j,s,t}(x_i^{p^s}+y_i^{p^s})(x_j^{p^t}+y_j^{p^t})+\sum_{i=1}^{n}\sum_{s=0}^{k-1}b_{l,i,s}(x_i^{p^s}+y_i^{p^s})+c_l\\
&=\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{s=0}^{k-1}\sum_{t=0}^{k-1}a_{l,i,j,s,t}(x_i^{p^s}x_j^{p^t}+x_i^{p^s}y_j^{p^t}+y_i^{p^s}x_j^{p^t}+y_i^{p^s}y_j^{p^t})+\sum_{i=1}^{n}\sum_{s=0}^{k-1}b_{l,i,s}(x_i^{p^s}+y_i^{p^s})+c_l\\
&=f_l(x_1,\cdots,x_n)+f_l(y_1,\cdots,y_n)+\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{s=0}^{k-1}\sum_{t=0}^{k-1}a_{l,i,j,s,t}\underline{(x_i^{p^s}y_j^{p^t}+y_i^{p^s}x_j^{p^t})}-f_l(0,\cdots,0)
\end{aligned}$$

…（２６）

【０３７８】

このような理由から、上記の第１及び第２実施形態に係る効率的なアルゴリズムを拡張し、上記の式（２３）で表現されるような３次以上の多変数多項式を公開鍵として利用するアルゴリズムを構築することが可能であると言える。

【０３７９】

ここで、上記の式（２２）で表現される多変数多項式（以下、２次多項式）と、上記の式（２３）で表現される多変数多項式（以下、多次多項式）との関係について考察しておきたい。いま、位数ｑ＝ｐの体上で定義されるｎｋ変数の２次多項式、及び位数ｑ＝ｐ^{k}の体上で定義されるｎ変数の多次多項式について考える。この場合、ｍｋ本の２次多項式で構成される連立方程式の解答困難性は、ｍ本の多次多項式で構成される連立方程式の解答困難性と等価である。例えば、位数２の体上で定義される８０本の８０変数２次多項式により構成される連立方程式と、位数２^{8}の体上で定義される１０本の１０変数多次多項式とは、その解答困難性が等価である。

【０３８０】

つまり、ＧＦ（ｐ^{k}）の要素と、ＧＦ（ｐ）^{k}の要素とを同型写像により同一視した場

合、位数ｑ＝ｐの体上で定義されるｍｋ本のｎｋ変数２次多項式の組で表現される関数と等価な、位数ｑ＝ｐ^{k}の体上で定義されるｍ本のｎ変数多次多項式の組で表現される関数が存在する。例えば、ＧＦ（２^{8}）の要素と、ＧＦ（２）^{8}の要素とを同型写像により同一視した場合、位数２の体上で定義される８０本の８０変数２次多項式の組で表現される関数と等価な、位数２^{8}の体上で定義される１０本の１０変数多次多項式の組で表現される関数が存在する。このような理由から、上記の２次多項式を利用するか、上記の多次多項式を利用するかは、任意に選択することが可能である。

【０３８１】

ここで、上記の２次多項式を利用する場合の計算効率と上記の多次多項式を利用する場合の計算効率について考察しておきたい。

【０３８２】

位数２の体上で定義されるｎｋ変数２次多項式を利用する場合、アルゴリズムに含まれる演算は、ｎｋ個の１ビット変数を対象に実行されることになる。つまり、演算の単位が１ビットとなる。一方、位数２^{k}の体上で定義されるｎ変数多次多項式を利用する場合、アルゴリズムに含まれる演算は、ｎ個のｋビット変数を対象に実行されることになる。つまり、演算の単位がｋビットとなる。このｋ（ｋ＝２，３，４，…）は任意に設定することができる。そのため、実装上、都合の良い値にｋを設定することにより、計算効率を向上させることが可能になる。例えば、３２ビットのアーキティクチャ上にアルゴリズムを実装する場合、１ビット単位で演算を実行する構成にするよりも、３２ビット単位で演算を実行する構成にする方が高い計算効率を得られる。

【０３８３】

このように、上記の第１及び第２実施形態に係る効率的なアルゴリズムを、多次多項式を公開鍵として利用できるように拡張することで、演算の単位を実装するアーキティクチャに合わせることが可能になる。その結果、計算効率を向上させることが可能になる。

【０３８４】

＜５：ハードウェア構成例＞

上記の各アルゴリズムは、例えば、図２６に示す情報処理装置のハードウェア構成を用いて実行することが可能である。つまり、当該各アルゴリズムの処理は、コンピュータプログラムを用いて図２６に示すハードウェアを制御することにより実現される。なお、このハードウェアの形態は任意であり、例えば、パーソナルコンピュータ、携帯電話、ＰＨＳ、ＰＤＡ等の携帯情報端末、ゲーム機、接触式又は非接触式のＩＣチップ、接触式又は非接触式のＩＣカード、又は種々の情報家電がこれに含まれる。但し、上記のＰＨＳは、Ｐｅｒｓｏｎａｌ　Ｈａｎｄｙ－ｐｈｏｎｅ　Ｓｙｓｔｅｍの略である。また、上記のＰＤＡは、Ｐｅｒｓｏｎａｌ　Ｄｉｇｉｔａｌ　Ａｓｓｉｓｔａｎｔの略である。

【０３８５】

図２６に示すように、このハードウェアは、主に、ＣＰＵ９０２と、ＲＯＭ９０４と、ＲＡＭ９０６と、ホストバス９０８と、ブリッジ９１０と、を有する。さらに、このハードウェアは、外部バス９１２と、インターフェース９１４と、入力部９１６と、出力部９１８と、記憶部９２０と、ドライブ９２２と、接続ポート９２４と、通信部９２６と、を有する。但し、上記のＣＰＵは、Ｃｅｎｔｒａｌ　Ｐｒｏｃｅｓｓｉｎｇ　Ｕｎｉｔの略である。また、上記のＲＯＭは、Ｒｅａｄ　Ｏｎｌｙ　Ｍｅｍｏｒｙの略である。そして、上記のＲＡＭは、Ｒａｎｄｏｍ　Ａｃｃｅｓｓ　Ｍｅｍｏｒｙの略である。

【０３８６】

ＣＰＵ９０２は、例えば、演算処理装置又は制御装置として機能し、ＲＯＭ９０４、ＲＡＭ９０６、記憶部９２０、又はリムーバブル記録媒体９２８に記録された各種プログラムに基づいて各構成要素の動作全般又はその一部を制御する。ＲＯＭ９０４は、ＣＰＵ９０２に読み込まれるプログラムや演算に用いるデータ等を格納する手段である。ＲＡＭ９０６には、例えば、ＣＰＵ９０２に読み込まれるプログラムや、そのプログラムを実行する際に適宜変化する各種パラメータ等が一時的又は永続的に格納される。

【０３８７】

これらの構成要素は、例えば、高速なデータ伝送が可能なホストバス９０８を介して相互に接続される。一方、ホストバス９０８は、例えば、ブリッジ９１０を介して比較的データ伝送速度が低速な外部バス９１２に接続される。また、入力部９１６としては、例えば、マウス、キーボード、タッチパネル、ボタン、スイッチ、及びレバー等が用いられる。さらに、入力部９１６としては、赤外線やその他の電波を利用して制御信号を送信することが可能なリモートコントローラ（以下、リモコン）が用いられることもある。

【０３８８】

出力部９１８としては、例えば、ＣＲＴ、ＬＣＤ、ＰＤＰ、又はＥＬＤ等のディスプレイ装置、スピーカ、ヘッドホン等のオーディオ出力装置、プリンタ、携帯電話、又はファクシミリ等、取得した情報を利用者に対して視覚的又は聴覚的に通知することが可能な装置である。但し、上記のＣＲＴは、Ｃａｔｈｏｄｅ　Ｒａｙ　Ｔｕｂｅの略である。また、上記のＬＣＤは、Ｌｉｑｕｉｄ　Ｃｒｙｓｔａｌ　Ｄｉｓｐｌａｙの略である。そして、上記のＰＤＰは、Ｐｌａｓｍａ　ＤｉｓｐｌａｙＰａｎｅｌの略である。さらに、上記のＥＬＤは、Ｅｌｅｃｔｒｏ－Ｌｕｍｉｎｅｓｃｅｎｃｅ　Ｄｉｓｐｌａｙの略である。

【０３８９】

記憶部９２０は、各種のデータを格納するための装置である。記憶部９２０としては、例えば、ハードディスクドライブ（ＨＤＤ）等の磁気記憶デバイス、半導体記憶デバイス、光記憶デバイス、又は光磁気記憶デバイス等が用いられる。但し、上記のＨＤＤは、Ｈａｒｄ　Ｄｉｓｋ　Ｄｒｉｖｅの略である。

【０３９０】

ドライブ９２２は、例えば、磁気ディスク、光ディスク、光磁気ディスク、又は半導体メモリ等のリムーバブル記録媒体９２８に記録された情報を読み出し、又はリムーバブル記録媒体９２８に情報を書き込む装置である。リムーバブル記録媒体９２８は、例えば、ＤＶＤメディア、Ｂｌｕ－ｒａｙメディア、ＨＤ　ＤＶＤメディア、各種の半導体記憶メディア等である。もちろん、リムーバブル記録媒体９２８は、例えば、非接触型ＩＣチップを搭載したＩＣカード、又は電子機器等であってもよい。但し、上記のＩＣは、Ｉｎｔｅｇｒａｔｅｄ　Ｃｉｒｃｕｉｔの略である。

【０３９１】

接続ポート９２４は、例えば、ＵＳＢポート、ＩＥＥＥ１３９４ポート、ＳＣＳＩ、ＲＳ－２３２Ｃポート、又は光オーディオ端子等のような外部接続機器９３０を接続するためのポートである。外部接続機器９３０は、例えば、プリンタ、携帯音楽プレーヤ、デジタルカメラ、デジタルビデオカメラ、又はＩＣレコーダ等である。但し、上記のＵＳＢは、Ｕｎｉｖｅｒｓａｌ　Ｓｅｒｉａｌ　Ｂｕｓの略である。また、上記のＳＣＳＩは、Ｓｍａｌｌ　Ｃｏｍｐｕｔｅｒ　Ｓｙｓｔｅｍ　Ｉｎｔｅｒｆａｃｅの略である。

【０３９２】

通信部９２６は、ネットワーク９３２に接続するための通信デバイスであり、例えば、有線又は無線ＬＡＮ、Ｂｌｕｅｔｏｏｔｈ（登録商標）、又はＷＵＳＢ用の通信カード、光通信用のルータ、ＡＤＳＬ用のルータ、又は接触又は非接触通信用のデバイス等である。また、通信部９２６に接続されるネットワーク９３２は、有線又は無線により接続されたネットワークにより構成され、例えば、インターネット、家庭内ＬＡＮ、赤外線通信、可視光通信、放送、又は衛星通信等である。但し、上記のＬＡＮは、Ｌｏｃａｌ　Ａｒｅａ　Ｎｅｔｗｏｒｋの略である。また、上記のＷＵＳＢは、Ｗｉｒｅｌｅｓｓ　ＵＳＢの略である。そして、上記のＡＤＳＬは、Ａｓｙｍｍｅｔｒｉｃ　Ｄｉｇｉｔａｌ　Ｓｕｂｓｃｒｉｂｅｒ　Ｌｉｎｅの略である。

【０３９３】

＜６：まとめ＞

最後に、本発明の実施形態に係る技術内容について簡単に纏める。ここで述べる技術内容は、例えば、ＰＣ、携帯電話、携帯ゲーム機、携帯情報端末、情報家電、カーナビゲーションシステム等、種々の情報処理装置に対して適用することができる。

【０３９４】

上記の情報処理装置の機能構成は次のように表現することができる。当該情報処理装置は、次のような鍵設定部と、メッセージ送信部と、検証パターン受信部と、回答送信部とを有する。上記の鍵設定部は、$s \in K^n$を秘密鍵に設定し、環K上の多次多項式$f_i(x_1, \cdots, x_n)$（$i=1 \sim m$）及び$y_i = f_i(s)$を公開鍵に設定するものである。$i=1 \sim m$について$y_i = f_i(s)$を求める問題は、多次多変数連立方程式の求解問題に他ならない。つまり、上記の情報処理装置は、多次多変数連立方程式の求解問題に安全性の根拠をおく認証の仕組みを提供するものである。

【０３９５】

また、上記のメッセージ送信部は、検証者に対してメッセージｃを送信するものである。さらに、上記の検証パターン受信部は、１つの前記メッセージｃに対するｋ通り（ｋ≧３）の検証パターンの中から前記検証者により選択された１つの検証パターンの情報を受信するものである。そして、上記の回答送信部は、ｋ通りの回答情報の中から、前記検証パターン受信部により受信された検証パターンの情報に対応する回答情報を前記検証者に送信するものである。但し、前記回答情報は、前記ｋ通りの回答情報を用いて実施した前記メッセージｃに対するｋ通りの検証パターンが全て成功した場合に秘密鍵ｓが計算可能となる情報である。

【０３９６】

上記のメッセージｃには、例えば、３つのメッセージc_1、c_2、c_3が含まれている。そして、各検証パターンでは、c_1、c_2、c_3のいずれか２つに対する検証が行われる。また、上記の回答情報は、各検証パターンに応じて、上記いずれか２つのメッセージに対する検証を行うために必要な情報である。但し、上記のメッセージｃを検証者に渡しても、秘密鍵の情報が一切漏れないことが１つの条件である。もう１つの条件は、メッセージｃと回答情報が揃っても秘密鍵の情報が一切漏れないことである。これらの条件を満たすメッセージｃの構成、及び回答情報の構成については、図４などを参照しながら、既に具体的に説明した通りである。

【０３９７】

（備考）

上記の鍵生成アルゴリズムＧｅｎは、鍵設定部の一例である。上記の証明者アルゴリズムＰは、メッセージ送信部、検証パターン受信部、回答送信部、応答受信部、多項式生成部、多項式送信部の一例である。上記の情報σは、回答情報の一例である。上記の検証者アルゴリズムＶは、メッセージ受信部、検証パターン選択部、検証パターン送信部、回答受信部、検証部、応答送信部、多項式受信部の一例である。上記の署名生成アルゴリズムＳｉｇは、メッセージ生成部、検証パターン選択部、署名生成部の一例である。

【０３９８】

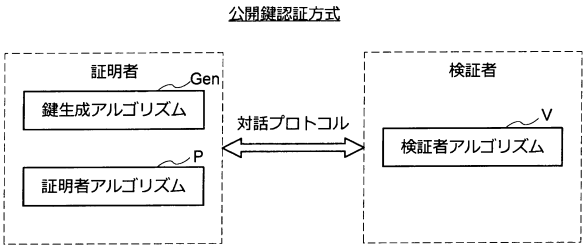
以上、添付図面を参照しながら本発明の好適な実施形態について説明したが、本発明は係る例に限定されないことは言うまでもない。当業者であれば、特許請求の範囲に記載された範疇内において、各種の変更例または修正例に想到し得ることは明らかであり、それらについても当然に本発明の技術的範囲に属するものと了解される。

【符号の説明】

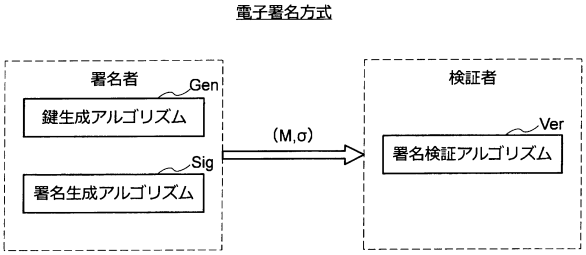
【０３９９】

Ｇｅｎ　鍵生成アルゴリズム
Ｐ　証明者アルゴリズム
Ｖ　検証者アルゴリズム
Ｓｉｇ　署名生成アルゴリズム
Ｖｅｒ　署名検証アルゴリズム

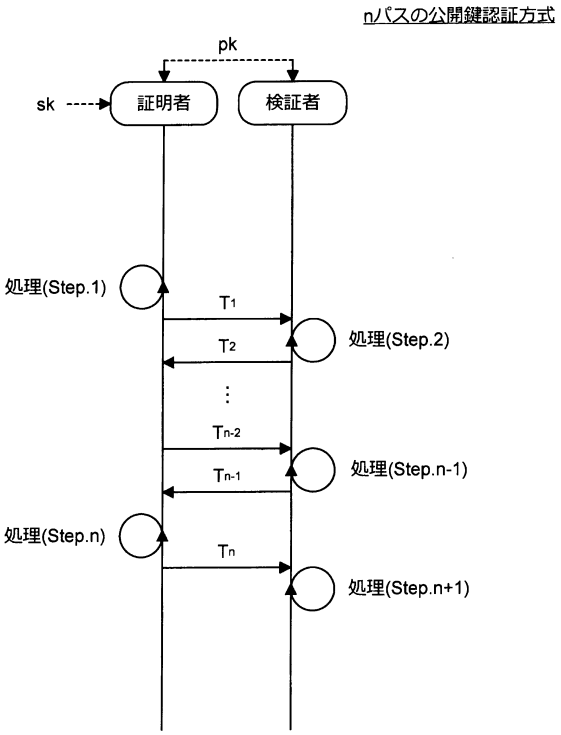
【図１】


公開鍵認証方式
証明者
Gen
鍵生成アルゴリズム
P
証明者アルゴリズム
対話プロトコル
検証者
V
検証者アルゴリズム

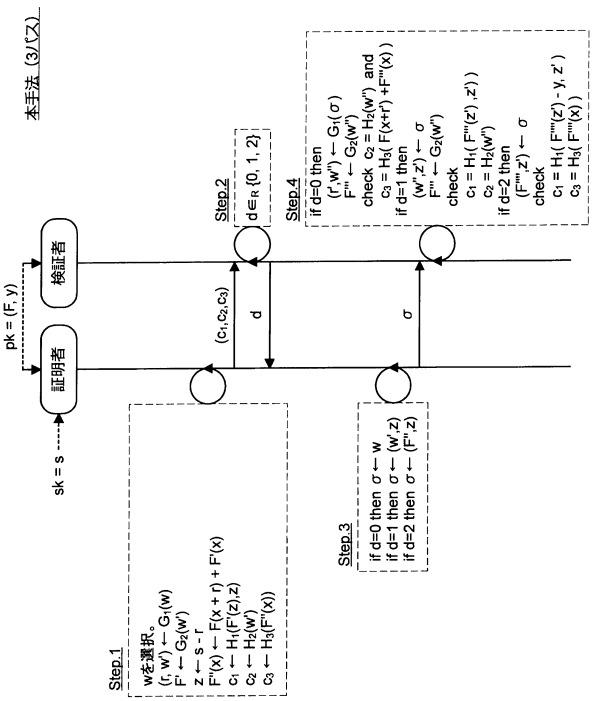
【図２】


電子署名方式
署名者
Gen
鍵生成アルゴリズム
Sig
署名生成アルゴリズム
(M,σ)
検証者
Ver
署名検証アルゴリズム

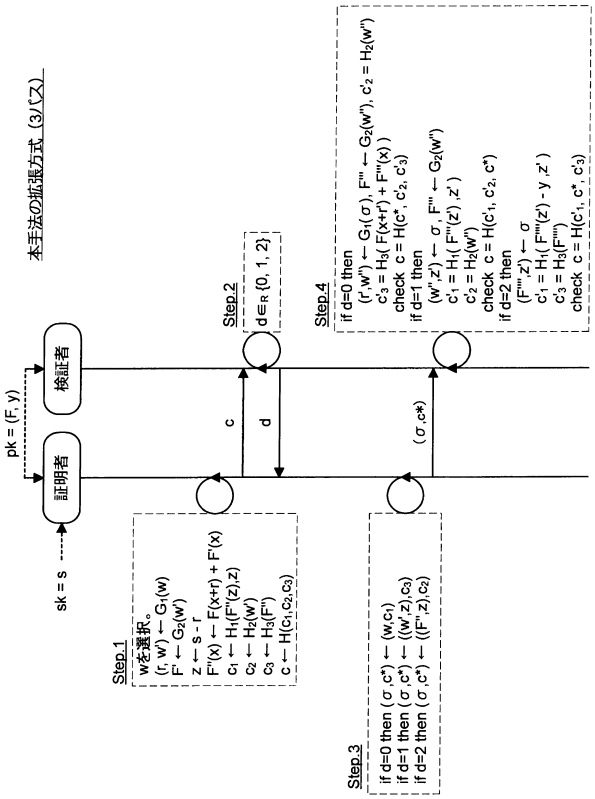
【図３】


nパスの公開鍵認証方式
pk
sk
証明者
検証者
処理(Step.1)
T1
処理(Step.2)
T2
Tn-2
処理(Step.n-1)
Tn-1
処理(Step.n)
Tn
処理(Step.n+1)

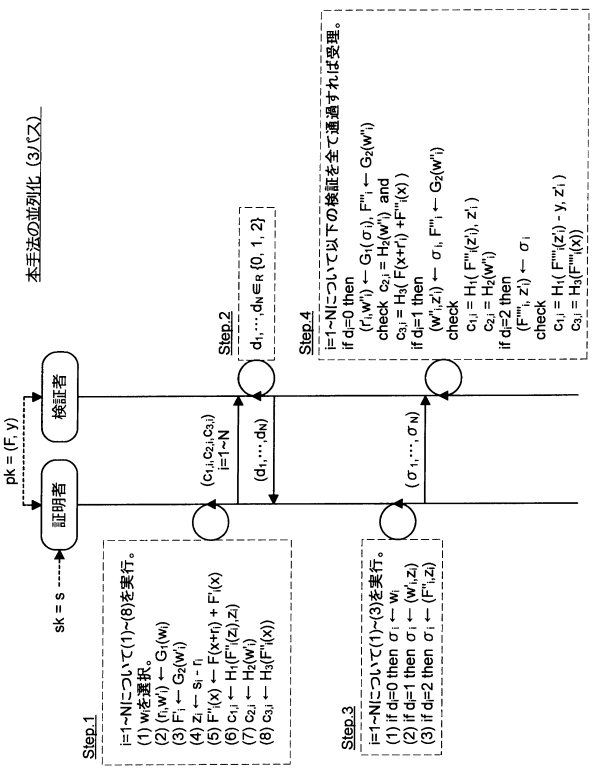
【図４】


本手法（3パス）
pk = (F, y)
sk = s
検証者
証明者
Step.1
Step.2
Step.3
Step.4
d
σ
(c1,c2,c3)

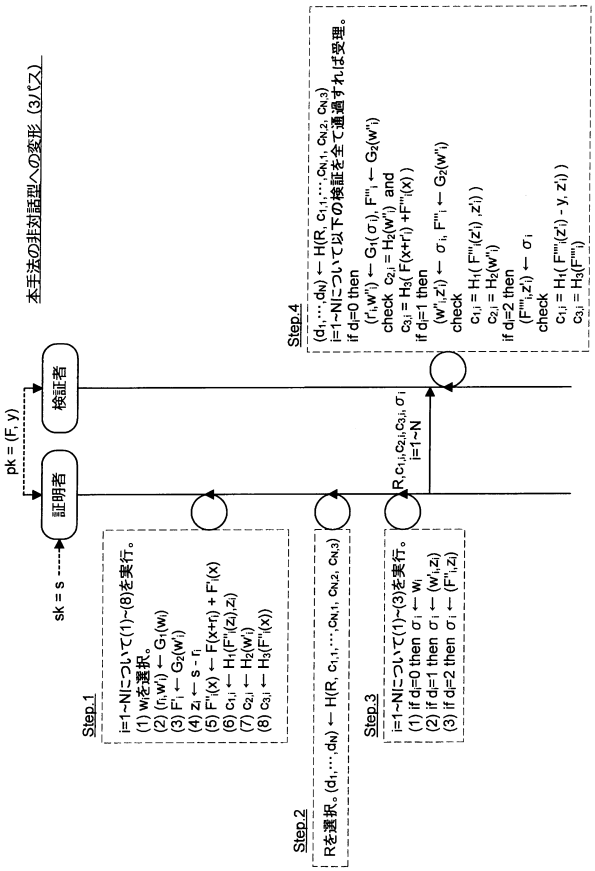
【図５】


本手法の拡張方式（3パス）
pk = (F, y)
sk = s
検証者
証明者
Step.1
Step.2
Step.3
Step.4
c
d
(σ,c*)

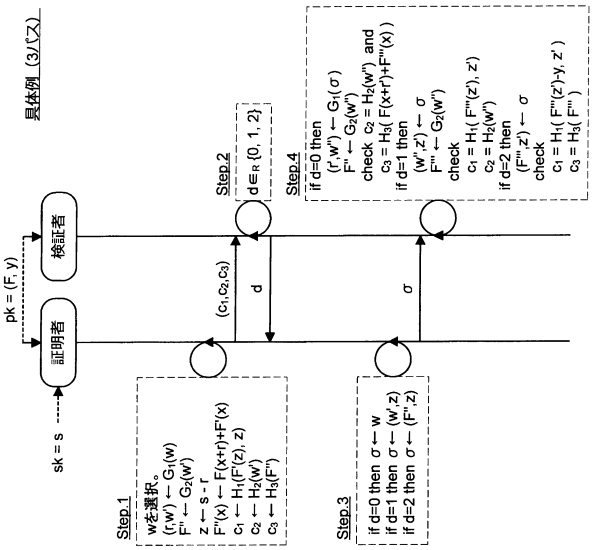
【図６】


本手法の並列化（3パス）
pk = (F, y)
sk = s
証明者
検証者
(c1,j,c2,j,c3,j) j=1~N
(d1,…,dN)
(σ1,…,σN)
Step.1
i=1~Nについて(1)~(8)を実行。
(1) wを選択。
(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(z),zi)
(7) c2,i ← H2(w'i)
(8) c3,i ← H3(F''i(x))
Step.2
d1,…,dN∈R {0, 1, 2}
Step.3
i=1~Nについて(1)~(3)を実行。
(1) if d=0 then σi ← wi
(2) if d=1 then σi ← (w'i,zi)
(3) if d=2 then σi ← (F''i,zi)
Step.4
i=1~Nについて以下の検証を全て通過すれば受理。
if d=0 then
(r'i,w'i) ← G1(σi), F'''i ← G2(w'i)
check c2,i = H2(w'i) and
c3,i = H3(F(x+r'i) +F'''i(x))
if d=1 then
(w'i,z'i) ← σi
check
c1,i = H1(F'''i(z'i), z'i)
c2,i = H2(w'i)
if d=2 then
(F'''i, z'i) ← σi
check
c1,i = H1(F''''i(z'i) - y, z'i)
c3,i = H3(F''''i(x))

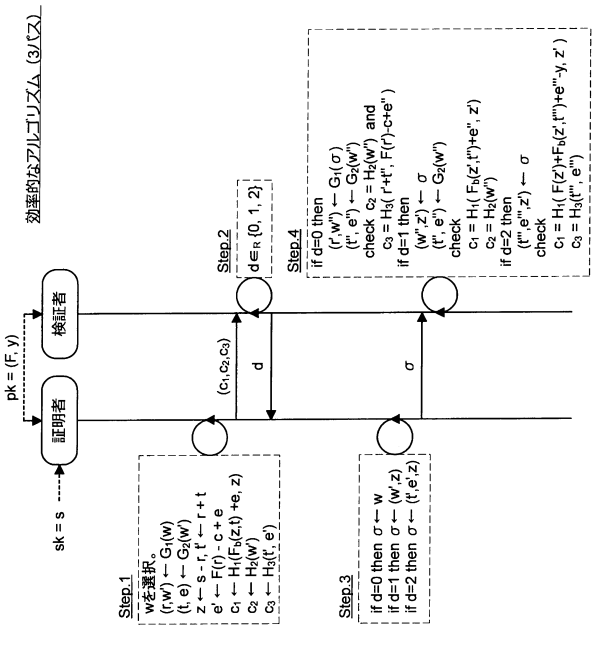
【図７】


本手法の非対話型への変形（3パス）
pk = (F, y)
sk = s
証明者
検証者
R,c1,i,c2,i,c3,i i=1~N
σi
Step.1
i=1~Nについて(1)~(8)を実行。
(1) wを選択。
(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(z),zi)
(7) c2,i ← H2(w'i)
(8) c3,i ← H3(F''i(x))
Step.2
R∈R {0, 1, 2}^N
Step.3
i=1~Nについて(1)~(3)を実行。
(1) if d=0 then σi ← wi
(2) if d=1 then σi ← (w'i,zi)
(3) if d=2 then σi ← (F''i,zi)
Step.4
(d1,…,dN) ← H(R, c1,1,…,cN,1, cN,2, cN,3)
i=1~Nについて以下の検証を全て通過すれば受理。
if d=0 then
(r'i,w'i) ← G1(σi), F'''i ← G2(w'i)
check c2,i = H2(w'i) and
c3,i = H3(F(x+r'i) +F'''i(x))
if d=1 then
(w'i,z'i) ← σi, F'''i ← G2(w'i)
check
c1,i = H1(F'''i(z'i), z'i))
c2,i = H2(w'i)
if d=2 then
(F'''i, z'i) ← σi
check
c1,i = H1(F''''i(z'i) - y, z'i))
c3,i = H3(F''''i)

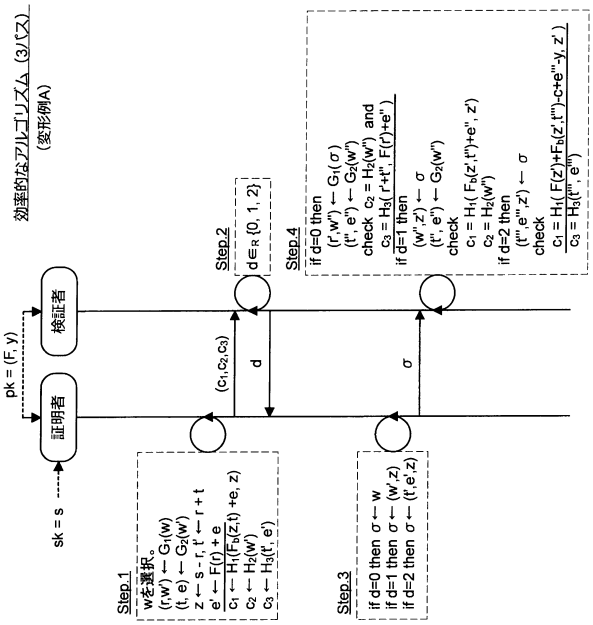
【図８】


具体例（3パス）
pk = (F, y)
sk = s
証明者
検証者
(c1,c2,c3)
d
σ
Step.1
wを選択。
(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'')
Step.2
d∈R {0, 1, 2}
Step.3
if d=0 then σ ← w
if d=1 then σ ← (w',z)
if d=2 then σ ← (F'',z)
Step.4
if d=0 then
(r',w'') ← G1(σ)
F'' ← G2(w'')
check c2 = H2(w'') and
c3 = H3(F(x+r')+F''(x))
if d=1 then
(w'',z') ← σ
F'' ← G2(w')
check
c1 = H1(F''(z'), z')
c2 = H2(w'')
if d=2 then
(F'',z') ← σ
check
c1 = H1(F''(z')-y, z')
c3 = H3(F'')

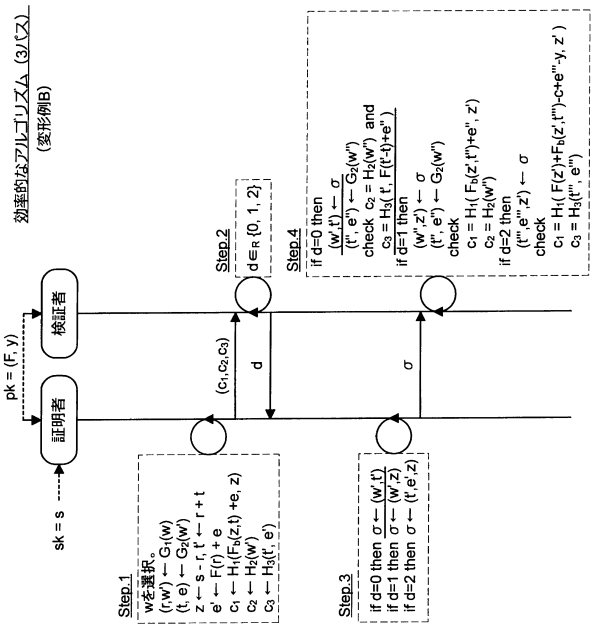
【図９】


効率的なアルゴリズム（3パス）
pk = (F, y)
sk = s
証明者
検証者
(c1,c2,c3)
d
σ
Step.1
wを選択。
(r,w') ← G1(w)
(t, e) ← G2(w')
z ← s - r, t' ← r + t
e' ← F(r) - c + e
c1 ← H1(F'(z,t) +e, z)
c2 ← H2(w')
c3 ← H3(t', e')
Step.2
d∈R {0, 1, 2}
Step.3
if d=0 then σ ← w
if d=1 then σ ← (w',z)
if d=2 then σ ← (t',e',z)
Step.4
if d=0 then
(r',w'') ← G1(σ)
(t', e') ← G2(w'')
check c2 = H2(w'') and
c3 = H3(r'+t", F(r')-c+e")
if d=1 then
(w",z') ← σ
(t", e') ← G2(w")
check
c1 = H1(F3(z',t")+e", z')
c2 = H2(w")
if d=2 then
(t",e",z') ← σ
check
c1 = H1(F(z')+F3(z',t")+e"-y, z')
c3 = H3(t", e")

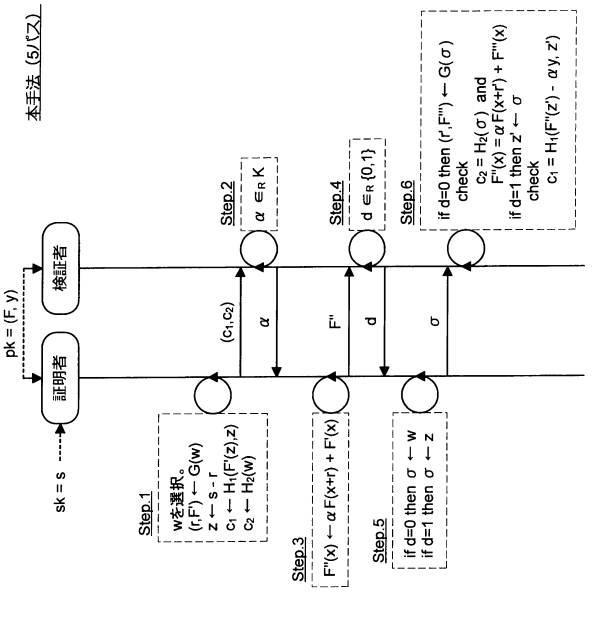
【図１０】



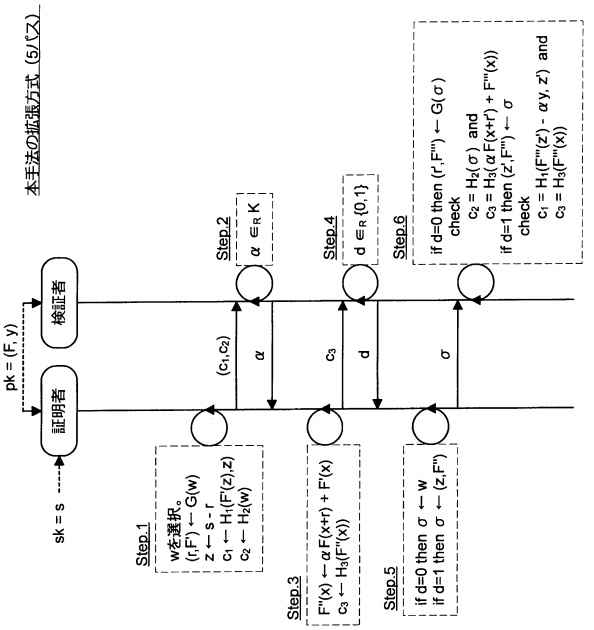
【図１１】



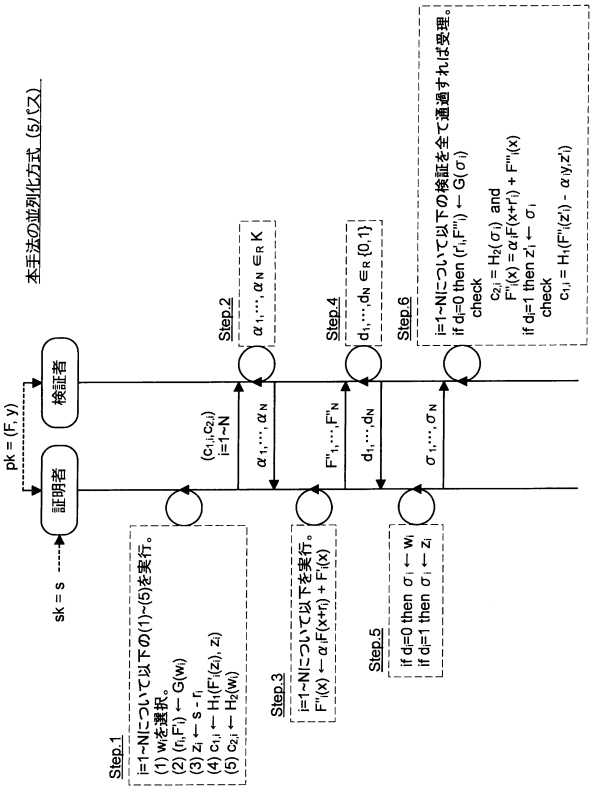
【図１２】



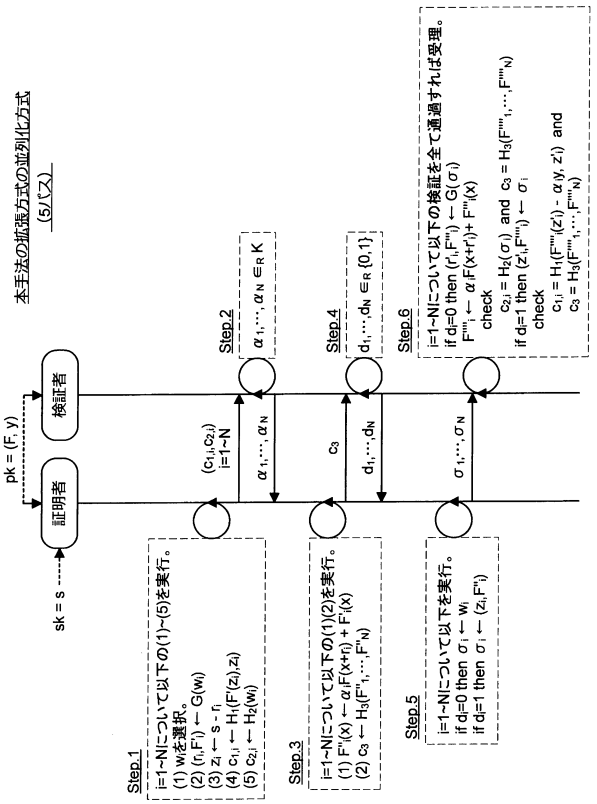
【図１３】



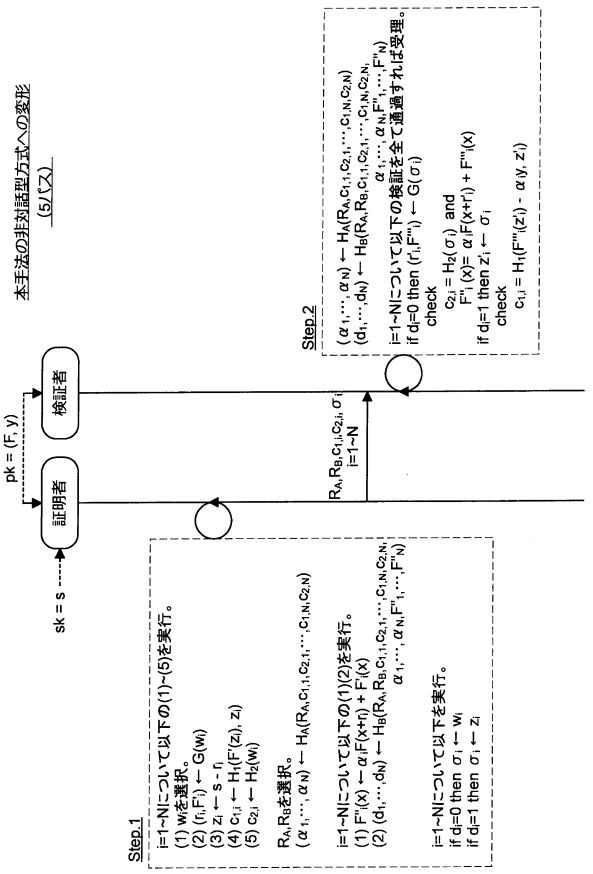
【図１４】



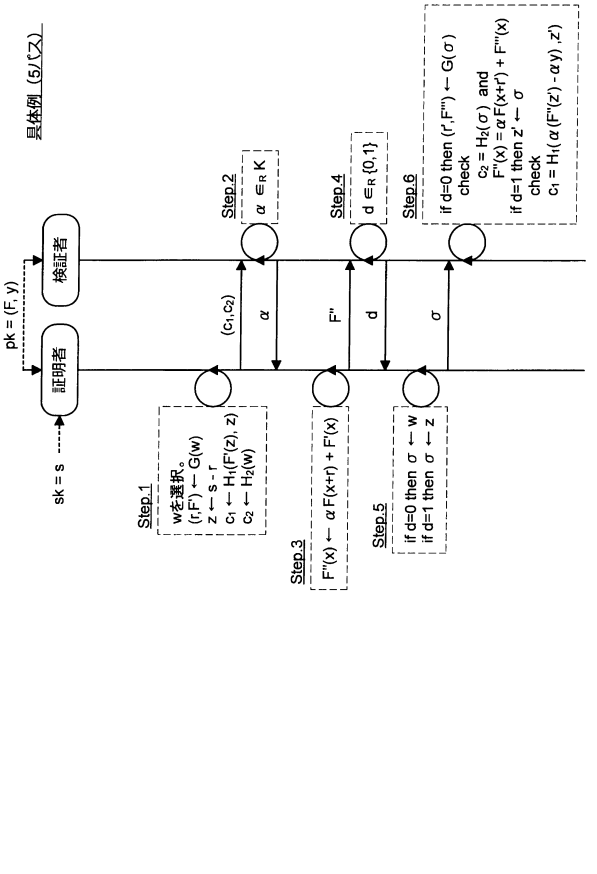
【図１５】



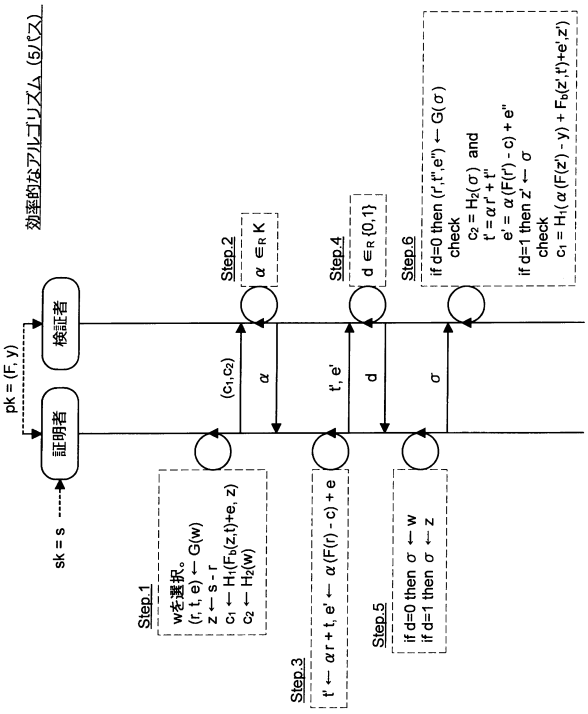
【図１６】



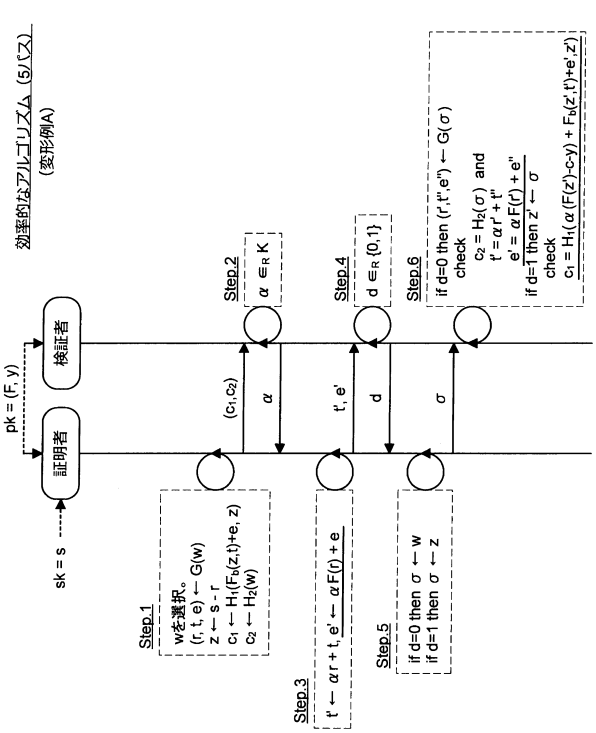
【図１７】



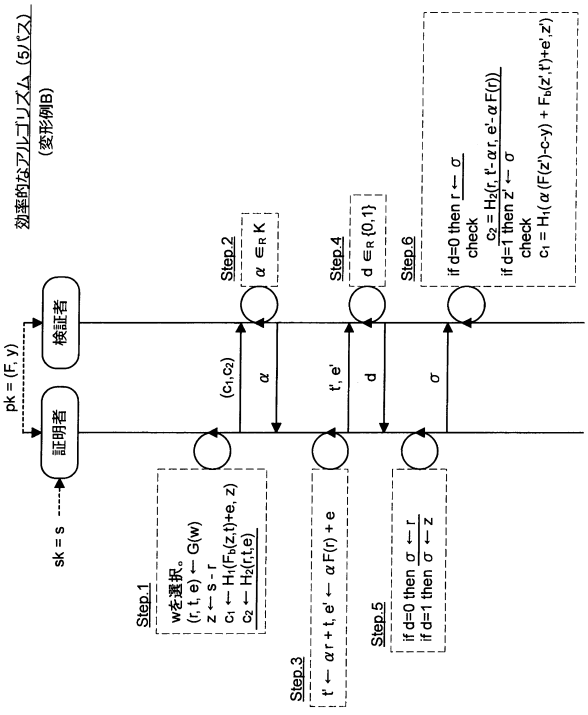
【図１８】



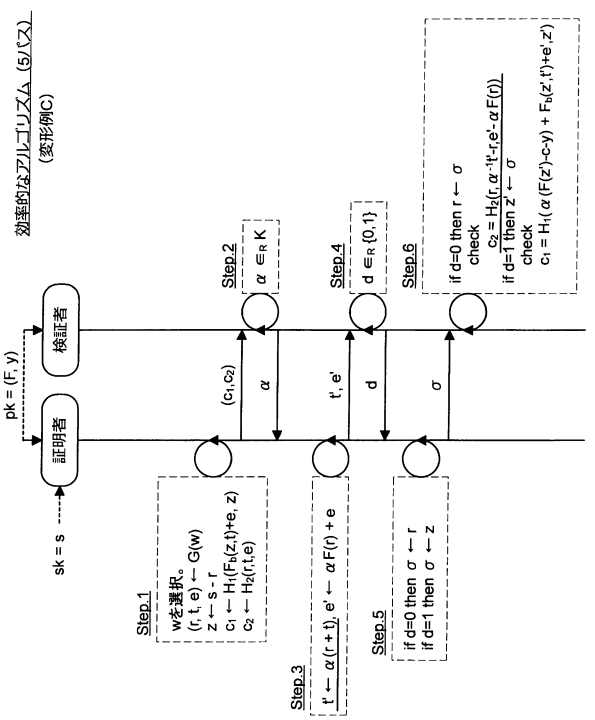
【図１９】



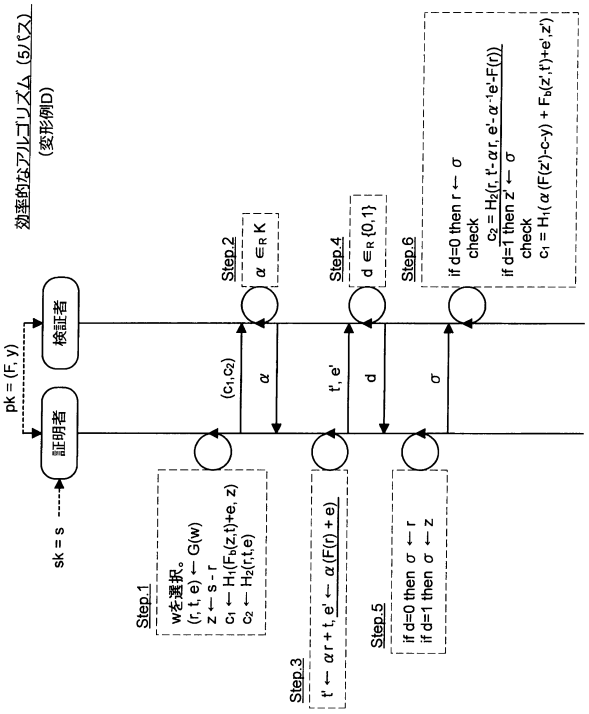
【図２０】



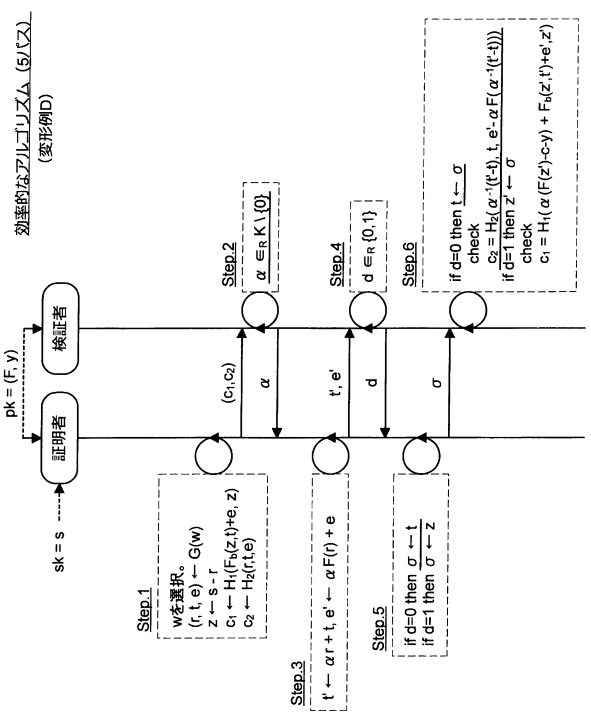
【図２１】



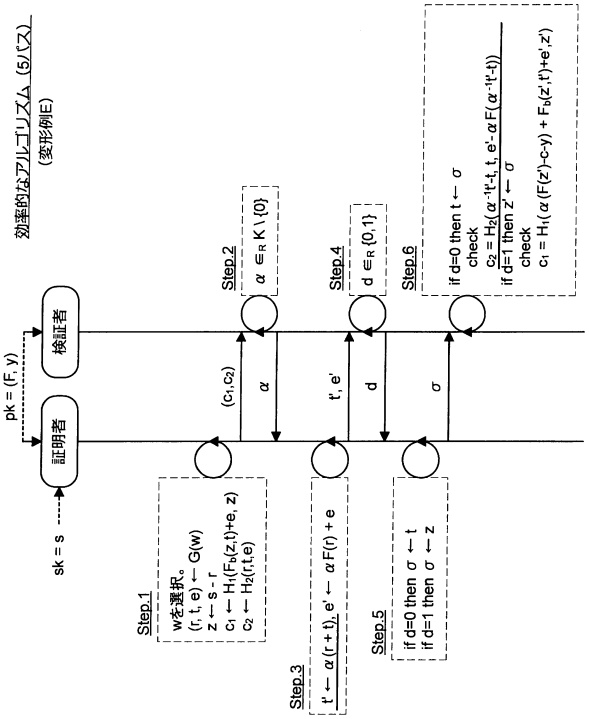
【図２２】



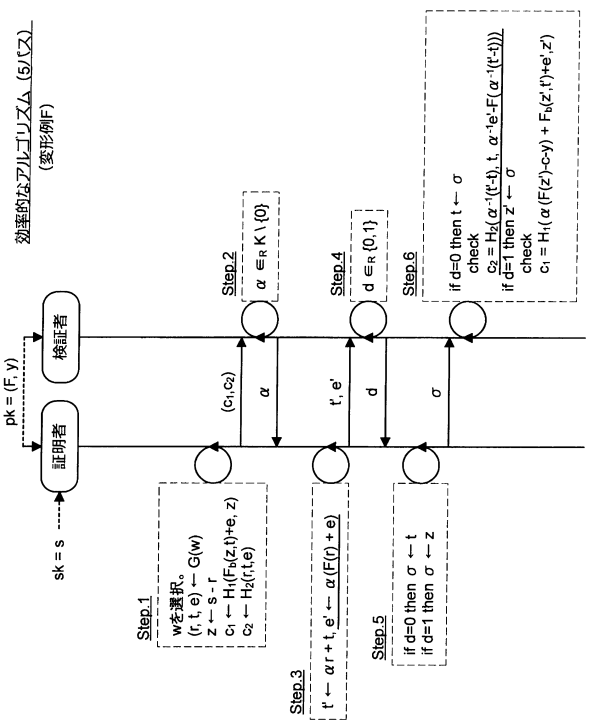
【図２３】



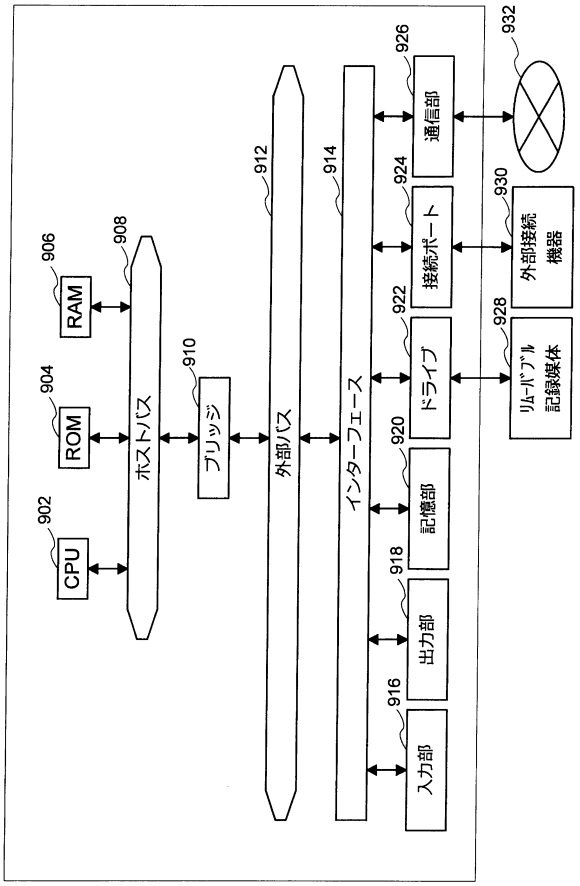
【図２４】



【図２５】



【図２６】

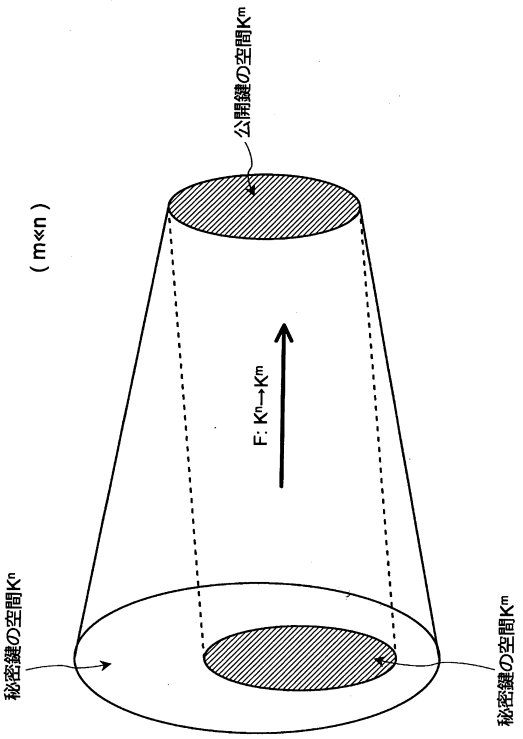


【図２７】

図表：3バスと5バスの効率比較

	3バス	5バス				
環の要素数q	–	$q = 2^4$	$q = 2^5$	$q = 2^6$	$q = 2^7$	$q = 2^8$
1回の偽証成功確率	0.667	0.531	0.516	0.508	0.504	0.502
n bit安全性に必要な繰り返し回数	1.710n	1.096n	1.046n	1.023n	1.011n	1.006n
80bit安全性に必要な繰り返し回数	137	88	84	82	81	81
128bit安全性に必要な繰り返し回数	219	141	134	131	130	129

【図２８】



フロントページの続き

(72)発明者　白井　太三
東京都港区港南１丁目７番１号　ソニー株式会社内
(72)発明者　樋渡　玄良
東京都港区港南１丁目７番１号　ソニー株式会社内

審査官　中里　裕正

(56)参考文献　特開平１１－２４９５６０（ＪＰ，Ａ）
特表２００５－５１５６５９（ＪＰ，Ａ）
PATARIN, J., Hidden Fields Equations(HFE) and Isomorphisms of Polynomials(IP): Two New Familes of Asymmetric Algotithms, Lecture Notes in Computer Science, １９９６年, Vol. 1070, p.33-48
KOMANO, Y. et al., ASS-CC: Provably Algebraic Surface Signature Scheme, 2010年暗号と情報セキュリティシンポジウム講演論文集, ２０１０年　1月１９日

(58)調査した分野(Int.Cl., ＤＢ名)
Ｈ０４Ｌ　　９／３２
ＪＳＴＰｌｕｓ／ＪＭＥＤＰｌｕｓ／ＪＳＴ７５８０（ＪＤｒｅａｍＩＩＩ）